# Interval Logic

— Proof Theory and Theorem Proving

**Thomas Marthedal Rasmussen**

# Interval Logic

— Proof Theory and Theorem Proving

**Thomas Marthedal Rasmussen**

PhD Thesis
Informatics and Mathematical Modelling
Technical University of Denmark
January 2002

This dissertation is submitted to Informatics and Mathematical Modelling (IMM) at the Technical University of Denmark in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

The work has been supervised by Associate Professor Michael R. Hansen, IMM, and Associate Professor Hans Rischel, IMM (now retired).

Kgs. Lyngby, January 31, 2002

Thomas Marthedal Rasmussen

# Abstract

*Real-time systems* are computer systems which have to meet real-time constraints. To increase the confidence in such systems, formal methods and formal verification are utilized. The class of logics known as *interval logics* can be used for expressing properties and requirements of real-time systems. By *theorem proving* we understand the activity of proving theorems of a logic with the assistance of a computer.

The goal of this thesis is to improve theorem proving support for interval logics such that larger and more realistic case-studies of real-time systems can be conducted using these formalisms. For achieving this goal we (1) investigate the foundations necessary for providing a useful theorem proving system for interval logics, and (2) actually provide such a system as well as conduct experiments with it.

We introduce an interval logic, *Signed Interval Logic* (SIL), which includes the notion of a direction of an interval, and present a *sound and complete Hilbert proof system* for it. Because of its generality, SIL can conveniently act as a *general formalism* in which other interval logics can be encoded.

We develop proof theory for SIL including both a *sequent calculus system* and a *labelled natural deduction system*. We conduct theoretical investigations of the systems with respect to subformula properties, proof search, etc.

The generic theorem proving system *Isabelle* is used as a framework for encoding both proof theoretical systems. We consider a number of examples/small case-studies and discuss strengths and weaknesses of the encodings.

From both a theoretical and a practical viewpoint, the labelled natural deduction system is the clear winner. We discuss how to scale the approach to larger case-studies.

# Resumé

*Tidstro systemer* er computer-systemer, som skal overholde visse tidskrav. Formelle metoder og formel verifikation kan bruges til at øge tilliden til sådanne systemer. Klassen af logikker betegnet som *intervallogikker* kan bruges til at udtrykke egenskaber ved, og krav til, tidstro systemer. Ved *bevisførelse* forstår vi den aktivitet at bevise sætninger i en logik ved hjælp af en computer.

Målet med denne afhandling er at forbedre bevisførerunderstøttelse for intervallogikker, således at større og mere realistiske case-studies for tidstro systemer kan bevises korrekte ved hjælp af disse formalismer. Vi vil, for at nå dette mål, (1) undersøge det nødvendige grundlag for at have at brugbart bevisførersystem, og (2) rent faktisk udvikle et sådant system og lave eksperimenter med det.

Vi introducerer en logik, *Signed Interval Logic* (SIL), som indeholder mulighed for at tale om retningen af et interval, og præsenterer et *sundt og fuldstændigt Hilbert bevis system* for den. SIL kan på grund af dens generalitet fungere som en *general formalisme*, som andre intervallogikker kan beskrives i.

Vi udvikler bevisteori for SIL i form af både et sekvent kalkyle ( *sequent calculus*) system og et mærket naturligt deduktions (*labelled natural deduction*) system. Vi foretager teoretiske undersøgelser af systemerne med henblik på delformelegenskaber, bevis-søgning, mv.

Det generiske bevisførersystem *Isabelle* bruges til at indkode begge bevisteoretiske systemer. Vi betragter en række eksempler/små case-studies og diskuterer stærke og svage sider ved indkodningerne.

Det mærkede naturlige deduktions system er den klare vinder set fra både et teoretisk og et praktisk synspunkt. Vi diskuterer hvordan bevisværktøjet kan skaleres til større case-studies.

# Acknowledgments

First and foremost, I would like to thank my main supervisor, Michael R. Hansen, for his support and encouragement over the years. He has always found the time for considering and discussing my ideas. Also thanks to my other supervisor, Hans Rischel, who unfortunately chose to retire during my PhD.

Part of the work reported in this thesis was carried out while I was visiting the Computer Laboratory at the University of Cambridge. I would like to thank Larry Paulson for inviting me and for providing valuable guidance on Isabelle and logic in general. Also thanks to my office mates, Dave Richerby and Giampaolo Bella, and all the others at the Lab who made my eight months stay in Cambridge a very pleasant experience.

Thanks to Sebastian Skalberg for becoming a good friend and for interesting discussions on logic, theorem proving and much more. I should in particular thank Sebastian for help on some technical aspects of Isabelle.

Torben Hoffman, followed by Morten P. Lindegaard, have been my office mates at IMM during the past years. I would like to thank them both for making the average day less boring and for comments and suggestions to my work.

Also thanks to Henrik Pilegaard, Mikael Buchholtz and Rene R. Hansen for creating a pleasant atmosphere on my floor and to all the other people at IMM who have helped me over the years, including our secretaries Maria Hansen and Eva Bing.

Finally, thanks to my brother Jan for proofreading a draft of the thesis and for help fine-tuning LaTeX.

# Contents

CHAPTER 1

# Introduction

*Real-time systems* are computer systems which have to meet real-time constraints: They have to react to events within a certain time interval, to produce output before a prescribed delay has elapsed, etc. Unfortunately, it is impossible to guarantee a correct behavior (in an absolute sense) of such systems. [1] This is of course a serious problem which can be nothing less than fatal in the case of safety-critical systems such as aircraft control systems and medico-technical equipment.

Although absolute correctness is impossible we can try to minimize the risk of failure considerable, in other words, to thrive for a high degree of confidence in the system. For this, formal methods are useful; they provide rigorous, mathematical frameworks and notations for expressing requirements to, and specifications of, systems. In the case of real-time systems, various real-time logics (cf., e.g., [AH91] for an overview of many such logics) have been proposed as basic formalisms. Real-time logics are (as the name suggests) logics which include notions for reasoning of time. *Interval logics* are examples of such logics; they will constitute the central logical formalism of this thesis and we will discuss them more thoroughly below.

Given a formal description of (a part of) a system we wish to somehow check that it satisfies the formal requirements for it. This process is known as *formal verification*. There are two major approaches to formal verification: *Theorem proving* and *model checking*. By theorem proving we understand the activity of using a computer for assistance when proving theorems of a particular logic (or particular logics). If the description of, and requirements to, a system are formulated within the same logical framework, this gives an approach to formal verification. This is exactly the approach to formal verification we will take in this thesis; we will discuss theorem proving and

[1] This is essentially the case for systems of any kind if they are just moderately complex.

theorem proving systems in more detail below.

Model checking (e.g., [CS01]) is the principle of modeling a system in terms of a (finite) state transition graph, formulate the requirements in an appropriate temporal logic, and then check that the model satisfies the temporal formula by, in principle, doing an exhaustive search of all reachable states. The main advantage of model checking is that it is fully automatic. On the other hand, it also has some limitations as it, e.g., risks suffering from state explosion problems.

In the case of real-time systems, the modeling is often done in terms of some kind of timed automata and a real-time logic is used for specifying requirements. A number of model checking tools for real-time system exist, e.g., UPPAAL [A+01] and KRONOS [Yov97].

We will not consider model checking further in this thesis, although we want to mention that various work on model checking where the requirements are specified in an interval logic, has been carried out as well, e.g., [Han94, ZZYL94, Frä97].

The goal of this thesis is to improve theorem proving support for interval logics such that larger and more realistic case-studies of real-time systems can be conducted using these formalisms.

For achieving this goal we wish to:

1. Investigate the foundations necessary for providing a useful theorem proving system for interval logics.
2. Actually provide such a system as well as conduct experiments with it.

The rest of this introduction is organized as follows: Section 1.1 provides some background to interval logic. In Section 1.2 we introduce proof theory and sketch its connection to interval logic and theorem proving. Then, in Section 1.3, we consider a number of actual theorem proving systems, leading to the system of choice in this thesis. Following this, in Section 1.4, we consider a couple of particular interval logics in greater detail. Thereafter, in Section 1.5, we discuss the main results of this thesis concerning proof theory and theorem proving for interval logics before concluding in Section 1.6. Finally, in Section 1.7 we give an overview of the rest of the thesis.

## 1.1 Interval Logic — Background

Historically, temporal logics have been considered since ancient times in connection with philosophical logic [ØH95]. In these temporal logics (in their most basic form), the truth-value of a formula is evaluated relative to a time-point, and it is possible to make qualitative reasoning concerning, e.g., all future time-points.

It is generally acknowledged that Pnueli [Pnu77] was the first to introduce temporal logic in computer science. The use of temporal logic in computer science is widespread today, e.g., within the model checking community.

But even though such temporal logics have turned out the most popular, variations of the theme have also been thoroughly considered, among these the topic of

this thesis: *Interval (temporal) logics*. In these logics, the truth-value of a formula is considered relative to a whole interval of time-points. Such temporal intervals are usually represented as pairs consisting of their beginning and end points, and can be based on discrete or dense time domains.

It is thus possible to express properties such as:

- "If property $\phi$ holds on this interval then property $\psi$ must hold on all subintervals."
- "Property $\chi$ must hold on some interval eventually."
- "If property $\xi$ holds on this interval then $\xi$ must hold on an interval immediately following this interval as well."

Interval logic also originates in philosophical logic but is much more recent [ØH95]. One of the first uses of an interval logic formalism in computer science was the work of [HMM83, Mos85] where timing aspects of hardware components were modeled. There has been a lot of work since, on many different aspects of interval logics, e.g., [Lew90, HS91, Ven91, Mos95, RM$^+$96a, RM$^+$96b].

### 1.1.1 Duration Calculus

In the ProCoS project [B$^+$89] in the end of the 1980's and the beginning of the 1990's it was realized that a convenient formalism for specifying and reasoning of accumulated durations of Boolean valued functions over time periods was required for expressing certain properties of real-time systems. This lead to the development of Duration Calculus (DC) [ZHR91] which is an extension of Interval Temporal Logic (ITL) [Mos85] with term-level notions for accumulated durations. The introduction of DC initiated much work on aspects of DC as well as, importantly, interval logics proper. As DC was introduced, the underlying ITL had not been thoroughly investigated; no complete proof system existed. This was not given until 1995 [Dut95a]. Together with a relative completeness result for DC and (un)decidability results this lead to [HZ97] which is the most comprehensive reference for the logical foundations of DC. Further work on (extended versions of) DC include [ZRH93, ZL94, LH94, ZHL95, Pan96].

### 1.1.2 Liveness

It soon became apparent that the original ITL had some limitations which made it difficult to specify (unbounded) liveness properties. An initial attempt to overcome this was [Ska94b]. Later, Neighbourhood Logic (NL) [ZH98] was introduced on a theoretically more firm basis. Most recently, Signed Interval Logic (SIL) [Ras99a] was proposed, with the introduction of the notion of a direction of an interval. SIL has (as ITL) only one interval modality but SIL is (contrary to ITL) capable of specifying liveness properties. Other interval logics capable of this (such as NL) have more than one interval modality.

In Section 1.4 we continue our discussion on interval logic.

## 1.2 Proof Theory and Automated Reasoning

*Proof theory* is (as the name suggests) the theory of proofs. As a mathematical discipline, it is part of mathematical logic and not much more than a century old. The original motivation for proof theory was to act as a tool for meta-logical investigations [HPJ00]. As such, the most influential formalism is probably that known as *Hilbert style* proof systems. In these systems, a few (simple) inference rules are included to tell how theorems can be combined to form new theorems. But the core of the system is a number of axioms stating "indisputable" truths, i.e., theorems that are "self-evident". Hilbert systems are often convenient for answering questions concerning completeness, consistency, etc.

Nonetheless, Hilbert systems are often (far) removed from the semantics (the intended meaning) of the logical system, thus making it hard to tell how proofs are actually constructed. This was part of the motivation for the introduction of *sequent calculus* and *natural deduction* systems by Gentzen [Gen35]. Consider, e.g., the familiar implication operator $\rightarrow$ of propositional logic. In natural deduction we have two rules involving $\rightarrow$:

$$\frac{\begin{array}{c}\alpha\\ \vdots\\ \beta\end{array}}{\alpha \rightarrow \beta} \qquad \frac{\alpha \rightarrow \beta \quad \alpha}{\beta} \quad .$$

The informal reading of $\rightarrow$ is here nicely reflected in the rules. The same is the case for the sequent calculus rules:

$$\frac{\Gamma, \alpha \vdash \beta, \Delta}{\Gamma \vdash \alpha \rightarrow \beta, \Delta} \qquad \frac{\Gamma \vdash \alpha, \Delta \quad \Gamma, \beta \vdash \Delta}{\Gamma, \alpha \rightarrow \beta \vdash \Delta} \quad .$$

As opposed to natural deduction, the assumptions ($\Gamma$) are made explicit in a sequent calculus.

With the appearance of computers, renewed interest in proof theory flourished. The main motivation was now to have proof theoretical formalisms suitable for implementation on a computer, in particular with the purpose of having the computer do the reasoning. The field of *automated reasoning* was started with the introduction of *resolution* in [Rob65] which initiated a considerable amount of research. Other formalisms aimed at automated reasoning include *semantic tableaux* [Smu68, Fit90] and *goal directed algorithmic proof theory* [Gab92].

For a general introduction to proof theory, see, e.g., [TS96]. Section 2 of the present thesis contains a technical discussion of basic proof theory for propositional and first order logic.

For the history of automated reasoning, see, e.g., [Dav01, Mac95]. For a contemporary and thorough survey of practically all aspects of automated reasoning, see [RV01].

### 1.2.1 Modal Logic

In *modal logic* the truth-value of a formula is evaluated relative to a current world, which belongs to a set of possible worlds. Hence, the truth-value of a formula can vary from one (possible) world to the next. In this context, interval logics can be seen as particular examples of modal logics by regarding the intervals as the possible worlds.

Proof theoretically, modal logics have traditionally been formulated using Hilbert style systems. Attempts have been made at defining natural deduction and sequent calculus systems for modal logics but this has often turned out difficult. Chapter 3 of this thesis is devoted to discussing this problem.

There have also been particular attempts from an automated reasoning perspective. A selection of the formalisms which have been modified for modal logic reasoning include: Resolution [Min90, HY99], semantic tableaux [Fit83], matrix methods [Wal90] (an extension of the semantic tableau formalism), goal directed algorithmic proof theory [GO00] and labelled deductive systems [Gab96, BMV97, BMV98a, Vig00]. The latter formalism plays a very central role in the present thesis.

## 1.3 Theorem Proving Systems

By *theorem proving system* we mean a system implemented on a computer with the purpose of assisting a user in proving theorems of a particular logic (or particular logics). We will in this section discuss a number of theorem proving systems.

There are interesting discussions on philosophical and historical aspects of the systems considered in this section in [Mac95, Har96]. Many of the systems discussed below are based on a version of (dependent) type theory. A contemporary discussion on the theoretical foundations of such systems can be found in [BG01] which also contains interesting comparisons of a number of these systems.

Theorem proving systems can (at least) be divided in three groups:

**Proof Checkers** Do exactly that: Check proofs. They provide a notation and framework for formulating proofs and checking their correctness but provide no means for automation. They can be thought of as advanced book-keeping systems.

The earliest and most influential of these systems was AutoMATH [NGdV94] developed by de Bruijn in the early 1970's. This system introduced many novel concepts which have inspired many later theorem proving systems.

Another interesting proof checking system is MIZAR [RT99] which is based on a version of set theory. In contrast to the other systems, it uses a fragment of natural language for both input and output of proofs.

**Automated Theorem Provers** The systems of this group aim (in principle) at being fully automatic, thus the user should just enter the formula to prove and then sit back and wait. Not surprisingly (depending on the problem area), this

is intractable when the problems get more complicated and the user thus have to guide the system somehow.

A successful system in this area is the Nqthm (Boyer-Moore) theorem prover [BM88]. The underlying logic is quantifier-free first order logic extended with recursive functions. This puts emphasis on proof by induction. Because of the power of the logic it has to be guided by a careful choice of lemmas. It thus has a steep learning curve but can be quite powerful for certain applications. A new and improved descendant of Nqthm is the ACL2 theorem prover [KMM00].

The most powerful automated theorem proving system today for first order logic with equality is probably Otter [Kal01] which is based on (advanced) resolution techniques and rewriting. It is the most truly automated theorem prover (of the systems considered here) but has because of its weak logic limited applications.

**Interactive Theorem Provers** These systems let the user interactively guide the proof construction. In their most basic form they act as proof checkers but they also contain facilities for automation.

One of the most popular of these systems is PVS [OSR93]. PVS is based on a version of higher-order logic with support for sub-typing. What makes PVS powerful is the use of various decision procedures for different logical domains. One of these is SVC [BDS96, BDS00] which is a powerful decision procedure for linear arithmetic. SVC is based on ideas of Shostak [Sho84, Sho78, Sho79, Sho77].

But what are probably the most popular kind of theorem proving systems today are those systems based on the LCF Approach. We will discuss this approach together with a number of systems below.

### 1.3.1 The LCF Approach

A question we have not considered so far is how we ensure the validity of a theorem proved by a theorem proving system. After all, software does not always behave as intended!

The most successful approach to this is often referred to as the "LCF approach" after the Edinburgh LCF system developed by Robin Milner and colleagues [Gor00]. The Edinburgh LCF system introduced many novel principles and has been very influential.

The main idea is to generate proofs in terms of extremely low-level primitive inferences, in order to provide a high level of assurance that the proofs are valid. The strict reduction to primitive terms is maintained by the abstract type system of the implementation language. (Often a version of the ML functional language). In other words, a small core provides a basic inference engine and as long as this core is correct no unsound derivations are possible no matter how complex the system gets. This should be contrasted to, e.g., the PVS system where bugs are still found now and then.

Based on the basic primitive rules, more and more substantial results are proved as derived rules, resulting in a body of readily applicable high-level rules. This works smoothly because of the type system. Rules are applied using tactics which can be combined by means of tacticals (programmable in ML).

Some of the most popular LCF-style systems are:

**Nuprl** [C$^+$86] is based on a version of Martin-Löf's Constructive Type Theory. The system is very large and complicated with the core containing more than 200 basic inference rules. It is possible to store proofs as proof objects.

**Coq** [Coq00] is constructive as Nuprl but based on a more contemporary theory, namely the Calculus of Constructions. Coq can (as Nuprl) manipulate proof objects.

**HOL** [GM93] is the most direct descendant of the Edinburgh LCF system. As opposed to Nuprl and Coq, HOL is based on a classical higher order logic.

**Isabelle** [Pau94] resembles HOL the most. The most distinguished feature of Isabelle is that it is generic. This means that it has a meta-logic acting as a logical framework in which various object logics can be encoded.

Isabelle is the system of choice in this thesis. The main reason for this is first and foremost the generic nature of Isabelle but also that it is one of the most developed theorem proving systems today.

In Chapter 7 we give a brief introduction to Isabelle.

## 1.4 Interval Logic — Technicalities

We now continue our discussion on interval logic. We become more technical and consider syntax and semantics of the interval logics we will be particularly interested in, namely Interval Temporal Logic (ITL) [Dut95a] and Signed Interval Logic (SIL) [Ras99a].

The syntax of ITL and SIL is basically the same, namely that of First Order Logic (FOL) with equality, with the addition of formulas built from the binary interval modality *chop*: $\frown$. We let $x, y, z, \ldots$ denote variables, $s, t, u, \ldots$ denote terms and $\phi, \psi, \chi, \ldots$ denote formulas. Hence, syntactically, we have formulas of the form $\phi \frown \psi$ besides the usual FOL formulas. Furthermore, both ITL and SIL include a special nullary function symbol $\ell$ which gives the *length* of an interval. This is the most distinguished feature of the interval logics we consider here compared to other kinds of interval logics.

Semantically, formulas of ITL are interpreted with respect to a given interval, which is represented by a pair $[i, j]$ (where $i \leq j$) of elements from an ordered temporal domain of time points. The meaning of the usual operators of FOL is independent of this interval whereas the meaning of $\frown$ is not; the semantics of $\frown$ is indicated in Figure 1.1. We will refer to $k$ of Figure 1.1 as the *chopping point* of $\frown$.

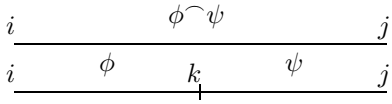



Figure 1.1: $\phi^\frown\psi$ holds on $[i,j]$ iff there is $k \in [i,j]$ such that $\phi$ holds on $[i,k]$ and $\psi$ on $[k,j]$.


Figure 1.2: $\Diamond_r\phi$ holds on $[i,j]$ iff there exists $k \geq j$ such that $\phi$ holds on $[j,k]$.

The chopping point will always lie inside the current interval on which we interpret a given formula. In general, modalities with this property are called *contracting*. With contracting modalities it is only possible to specify safety properties of a system. This is because once we have chosen the interval we want to observe, we are restricted to specifying properties of this interval and its subintervals.

To specify (unbounded) liveness properties, we need to reach intervals outside the current interval. In general, modalities which can do this are called *expanding*. Neighbourhood Logic (NL) [ZH98] is an example of an interval logic with expanding modalities. NL has two modalities, $\Diamond_r$ and $\Diamond_l$, for reaching a right neighbourhood and a left neighbourhood, respectively, of the current interval. This intuition is made more precise in Figure 1.2 in the case of $\Diamond_r$. The case of $\Diamond_l$ is similar.

SIL is an extension of ITL with the introduction of the notion of a direction (which can be either forward or backward) of an interval. The idea for SIL originates in [ER94] where an interval logic with such a notion of a direction of an interval was informally developed. An interval with a direction is in SIL represented by a *signed interval* $(i,j)$. Both the pair $(i,j)$ and the pair $(j,i)$ represent the same interval but $(j,i)$ has opposite direction of $(i,j)$. See Figure 1.3. In the figure, the direction of an interval is marked with a small arrowhead in either end of the interval.

In SIL, $\ell$ now gives the *signed length* of an interval. Intuitively, the absolute value of $\ell$ gives the length of the interval and the sign of $\ell$ determines the direction. Because

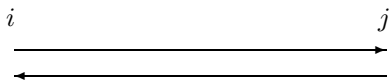



Figure 1.3: Both $(i,j)$ and $(j,i)$ represent the same interval but $(j,i)$ has opposite direction of $(i,j)$.

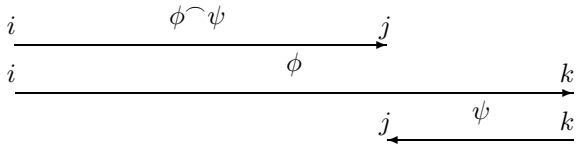



Figure 1.4: $\phi^\frown\psi$ holds on $(i,j)$ iff there is $k$ such that $\phi$ holds on $(i,k)$ and $\psi$ on $(k,j)$.

of the directions of intervals, the meaning of $^\frown$ in SIL is altered: See Figure 1.4. The chopping point can now lie anywhere and not just inside the current interval. This means that $^\frown$ of SIL has become an expanding modality, hence SIL can specify liveness properties.

ITL and SIL are modal logics. Formally, the semantics sketched above is given in terms of Kripke structures where the possible worlds are intervals. If we let $\mathfrak{M}$ be a first order Kripke model, the formal semantics of, e.g., $\wedge$ can be given as:

$$\mathfrak{M},(i,j) \models \alpha \wedge \beta \quad \text{iff} \quad \mathfrak{M},(i,j) \models \alpha \text{ and } \mathfrak{M},(i,j) \models \beta.$$

Hence, the semantics of $\wedge$ is independent of the interval $(i,j)$. Similarly for the other Boolean connectives. In the case of $^\frown$ of SIL we have:

$$\mathfrak{M},(i,j) \models \alpha^\frown\beta \quad \text{iff} \quad \mathfrak{M},(i,k) \models \alpha \text{ and } \mathfrak{M},(k,j) \models \beta \text{ for some } k.$$

In the case of $^\frown$ of ITL, the semantics is the same except that $k$ is constrained by $i \leq k \leq j$. The semantics of $\ell$ is given by a certain measure which of course is dependent on the given interval.

Proof theoretically, ITL and SIL have both been given sound and complete Hilbert style proof systems, see [Dut95a] and [Ras99a], respectively.

Chapter 4 contains an in-depth presentation and discussion of the topics of this section.

## 1.5 Interval Logic — Proof Theory and Theorem Proving

In this section we discuss the main topics of this thesis, namely proof theory and theorem proving for interval logics. We build on the background established in the previous sections.

We begin by discussing attempts at defining sequent calculus systems for interval logics as well as corresponding encodings in theorem proving systems. Despite some work on these subjects the results are not completely satisfactory. Thus, we following consider a proof theoretical framework which seems much more promising, namely that of Labelled Natural Deduction. We consider an encoding of this system in Isabelle and discuss some lessons learned from using it.

### 1.5.1 Sequent Calculus

What seems to have been the first work on theorem proving for interval logic was a semantic encoding of ITL and DC in PVS in 1994, giving PC/DC [Ska94a]. There various inference rules for ITL and DC are formulated in a sequent style. As mentioned in Section 1.1, at that time no complete proof system for ITL existed, hence the rules are somewhat ad hoc.

Later, based on the complete Hilbert system, an encoding of ITL and DC in Isabelle was carried out, giving Isabelle/DC [Hei99]. There, ITL and DC are encoded on top of the FOL sequent calculus LK of Isabelle. As a result, Isabelle/DC does not take much advantage of the sequent formalism: In essence, the axioms of the Hilbert system are added directly as axioms to LK, thus giving a mix of a Hilbert and a sequent system. As a consequence, the advantages of a sequent calculus system are not fully exploited.

In [Ras01a] a sequent calculus proof system for SIL is considered. Here an attempt is made to take advantage of the sequent calculus formalism as such and not just add axioms. This results in rules such as

$$\frac{\Gamma, \phi^\frown\chi \vdash \Delta \quad \Gamma, \psi^\frown\chi \vdash \Delta}{\Gamma, (\phi \vee \psi)^\frown\chi \vdash \Delta} \quad ,$$

which mimics the left-introduction rule for $\vee$ known from propositional logic but here "under the chop". There are more rules in the same style, resulting in a complete system. Because of the structure of the rules, it is possible to achieve a version of the subformula property for the system which is convenient for backwards proof search. *It is not a "proper" sequent calculus system though*, in that $^\frown$ does not appear in exactly one left- and one right-introduction rule. By these measures it is in fact not likely that a proper system exists at all; this seems to be the case for most modal logics in general [BS84] (cf. Section 1.2.1 as well). Despite this negative indication, it is seen how far the formalism can be "pushed": The main theoretical result is a "decidability modulo cut" result which entails that if one ignores the cut rule (which is necessary for completeness) provability is decidable (SIL is provably undecidable in general).

An encoding in Isabelle of this sequent calculus system has been carried out as well. Quite some automation support has been achieved for the encoding, including many rewrite rules and many derived rules used in search tactics — the latter taking advantage of the form of the sequent calculus system.

We refer to Chapters 5 and 8 for a more thorough discussion of the topics of this section.

### 1.5.2 Labelled Natural Deduction

In this section we consider Labelled Natural Deduction (LND) systems for interval logics. This proof theoretical framework is fundamentally different from the classical approaches: The intervals, which so far have appeared only in the semantics, are made

part of the syntax and thereby an important part of the proof system. This approach is inspired by work on Labelled Modal Logic [Vig00] which in turn was carried out in response to Gabbay's program on Labelled Deductive Systems [Gab96].

The most important consequence of the LND formalism is that it is *possible to have a "proper" natural deduction system* with exactly one introduction- and one elimination-rule for each connective — including the modalities. In the case of $^\frown$ of SIL we have the following rules:

$$\frac{(i,k):\phi \quad (k,j):\psi}{(i,j):\phi^\frown\psi}\ ^\frown I \qquad \frac{\begin{array}{c}(i,j):\phi^\frown\psi \qquad \begin{array}{c}[(i,k):\phi]\ [(k,j):\psi]\\ \vdots\\ (m,n):\chi\end{array}\end{array}}{(m,n):\chi}\ ^\frown E \quad .$$

In [Ras01d] a sound and complete LND system for SIL is given. The main theoretical result is a normalization result which implies that normal derivations satisfy a subformula property. In [Ras01c] it is furthermore discussed how the LND system for SIL can act as a general framework in which ITL and NL can be formulated.

An encoding of the LND system has been carried out in Isabelle. The encoding is close to the theoretical system. There were a few technical points to consider, e.g., how most conveniently to perform simplification on labelled formulas. The ability of the system to act as a general framework was also explored by encoding both ITL and NL in it.

We refer to Chapters 6 and 8 for a more thorough discussion of the topics of this section.

### 1.5.3 Examples and Experience

A number of examples have been conducted in both the sequent calculus and the LND encoding, primarily the latter, though, as it soon became apparent that the LND system was much more convenient. The three main reasons being:

- Reasoning in the LND system is much more intuitive; the intervals, which are part of the logic, can easily be visualized and the connection to the semantics is much clearer.
- A higher degree of automation is possible in the LND encoding; this fact owes a lot to the proper natural deduction system defined for SIL.
- Isabelle is inherently a system for doing reasoning in natural deduction systems; the sequent calculus encoding can seem less natural to use.

In conclusion, it is easier and more intuitive to conduct proofs in the LND system than the sequent calculus system, and the proof scripts are generally much shorter.

The concrete examples we have considered include small case-studies concerning simple properties of an oscillator, a gas burner and the deadline driven scheduler. The examples were discussed in [Ras01c] and are considered in further detail in Chapter 9.

## 1.6 Conclusion

The goal of this thesis was to improve theorem proving support for interval logics such that larger and more realistic case-studies of real-time systems can be conducted using these formalisms.

For this, we have investigated the following two main topics (in particular their interaction):

- Proof theory for interval logics.
- Encodings of interval logics in a theorem proving system (Isabelle).

Proof theoretically, we have investigated both a sequent calculus and a labelled natural deduction system for interval logic. We have tried to make the systems as suitable as possible for actual proof making.

Both systems have been encoded in Isabelle and examples have been conducted identifying strengths and weaknesses. From both a theoretical and a pragmatic viewpoint the undisputed "winner" was the LND system.

The fact that the labelled formalism turned out the most successful fits well with the ideas of the program of Gabbay [Gab96] on using labelled formalisms for non-classical logics. In fact, the basic idea of being able to refer to structures/elements, which previously only were considered part of the semantics of a logic, has really flourished the past 5–10 years, and is still a very popular topic, e.g., [GM02]. This is also reflected in the considerable interest in a general logical approach to this phenomenon known as Hybrid Logics, see, e.g., [Bla00].

We learned from the examples conducted in the LND encoding that we have a promising and convenient framework for doing interval logic theorem proving. The examples are all fairly small, though, so the question is, does the approach scale to larger examples and case-studies? Our claim is, yes, it does — if we "port" the system to the encoding of higher-order logic in Isabelle (Isabelle/HOL). This is because we need better support for arithmetic reasoning as well as possibilities to reason with sets, lists, etc., for the approach to scale. But this is exactly what Isabelle/HOL gives us and we should not expect the port to present any essential difficulties.

A good way of supporting this claim of scalability would be to mechanize the full proof of correctness of the Deadline Driven Scheduler (cf. Section 9.5). This is a large, rather complex case-study which would really exercise the interval logic encoding, include substantial arithmetic reasoning, and utilize the ability to reason with sets etc. These aspects are discussed in further detail in Chapter 9.

## 1.7 Organization

The present thesis is divided in nine chapters of which this introductory chapter is the first. Below we give an overview of chapters two through nine.

**Chapter 2** discusses proof theory for propositional and first order logic.

**Chapter 3** considers various modal logics and in particular proof theory for these.

**Chapter 4** gives a detailed introduction to the interval logics we will be particularly interested in as well as a number of related interval logics.

**Chapter 5** discusses a sequent calculus system for interval logic and considers pros and cons of the system.

**Chapter 6** considers a labelled natural deduction system for interval logic and discusses various extensions.

**Chapter 7** gives an introduction to the theorem proving system of choice in this thesis, namely Isabelle.

**Chapter 8** considers encodings of both the sequent calculus and the labelled natural deduction systems in Isabelle.

**Chapter 9** discusses a number of examples conducted in the encodings and give directions for future work.

CHAPTER 2

# Proof Theory

This chapter is concerned with introducing basic proof theoretical results for classical propositional and first order logic. The presentation will be fairly concise as we concentrate on introducing the relevant technical notions as well as fixing our notation. Hence, the reader is assumed to have prior basic knowledge of logic and proof theory.

Introductory texts on propositional and first order logic abound, e.g., [Smu68, Gal86, Men87, Ham88, Fit90]. Of these, [Gal86, Fit90] put emphasis on proof theory and automated reasoning. A good, general introduction to proof theory is [TS96]. For discussions on the history and philosophy of proof theory we refer to, e.g., [HPJ00].

Logics are often defined using *Hilbert style* proof systems (e.g., [Men87]). These proof systems often turn out the most convenient for reasoning *about* the logic. When reasoning *in* the logic is the main concern one often turns to *Sequent Calculi* or *Natural Deduction* systems, both of which were introduced by Gentzen [1] in [Gen35]. These systems are closer to semantic reasoning than Hilbert systems. Natural deduction has been thoroughly investigated by Prawitz in [Pra65]. A contemporary treatment of sequent calculi and natural deduction systems is given in [TS96].

## 2.1 Propositional Logic

### 2.1.1 Syntax

In *Propositional Logic* (PL), formulas $\alpha, \beta, \gamma, \ldots$ are constructed from an infinite set of *propositional letters* $p, q, r, \ldots$ and $\perp$ (denoting falsity) using the usual Boolean

[1] Sequent Calculi systems are also known as Gentzen systems.

operators.[2] Thus, propositional formulas are defined by the following abstract syntax:

$$\alpha \quad ::= \quad p \mid \bot \mid \neg\alpha \mid \alpha \wedge \beta \mid \alpha \vee \beta \mid \alpha \rightarrow \beta \mid \alpha \leftrightarrow \beta.$$

For convenience, we will assume the following precedence of the Boolean operators (highest first): 1) $\neg$, 2) $\wedge$, 3) $\vee$, 4) $\rightarrow, \leftrightarrow$. Formulas with no operators are called *atomic*. The *size* of a formula is the number of operators in the formula; thus, an atomic formula has size 0.

In PL we can talk of *functionally complete sets of operators*. This means that by taking a few operators as basic the rest can be defined in terms of those. Two often used functionally complete sets of operators for PL are $\{\neg, \wedge\}$ and $\{\bot, \rightarrow\}$. (In the latter case we can define $\neg\alpha \mathrel{\widehat{=}} \alpha \rightarrow \bot$.) We will in this thesis often restrict attention to such complete sets even if we do not mention it explicitly. When we use other operators they should thus be seen as defined abbreviations in the usual way.

### 2.1.2 Semantics

To define the semantics of PL we need a *valuation* $\mathcal{U}$ mapping propositional letters to the set of truth values $\{\mathsf{tt}, \mathsf{ff}\}$. We always have $\mathcal{U}(\bot) = \mathsf{ff}$ for any $\mathcal{U}$. Given a valuation $\mathcal{U}$ we inductively define *satisfaction* of a formula $\alpha$ (written $\mathcal{U} \models \alpha$) as follows:

$$\begin{array}{lll} \mathcal{U} \models p & \text{iff} & \mathcal{U}(p) = \mathsf{tt}, \\ \mathcal{U} \models \alpha \rightarrow \beta & \text{iff} & \mathcal{U} \models \alpha \text{ implies } \mathcal{U} \models \beta. \end{array}$$

Note how we utilize the fact that $\{\bot, \rightarrow\}$ is a functionally complete set of operators for PL. A corresponding (direct) formulation of the semantics for the remaining (defined) operators would be straightforward.

Given a formula $\alpha$, if $\mathcal{U} \models \alpha$ for all $\mathcal{U}$ we say that $\alpha$ is *valid* or a *tautology*, which is written $\models_{\mathrm{PL}} \alpha$.

### 2.1.3 Hilbert System

Hilbert proof systems for PL are fairly simple. One possible axiomatization is: [3]

P1: $\alpha \rightarrow (\beta \rightarrow \alpha)$,

P2: $(\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow ((\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma))$,

P3: $(\neg\beta \rightarrow \neg\alpha) \rightarrow ((\neg\beta \rightarrow \alpha) \rightarrow \beta)$,

together with the rule *modus ponens*:

MP: $\dfrac{\alpha \quad \alpha \rightarrow \beta}{\beta}$ .

[2] The symbol $\bot$ is a nullary operator but it is sometimes convenient to think of it as a propositional letter.

[3] Note that our formulation uses axiom schemata.

**Definition 2.1** A *proof* of a formula $\alpha$ is a finite sequence of formulas $\alpha_1, \ldots, \alpha_n$, where $\alpha_n$ is $\alpha$, and each $\alpha_i$ is either an instance of one of the above axioms or obtained by applying modus ponens to two previous members of the sequence. We write $\vdash_{\mathrm{PL}} \alpha$ to denote that there exists a proof of $\alpha$ in PL and we then say that $\alpha$ is a *theorem* of PL. We let $\nvdash_{\mathrm{PL}} \alpha$ denote that no proof of $\alpha$ exists, hence $\alpha$ is not a theorem of PL.

Given a set of formulas $\Gamma$, we say that there exists a *deduction of* $\alpha$ *from* $\Gamma$ (written $\Gamma \vdash_{\mathrm{PL}} \alpha$) if there is a sequence of formulas $\alpha_1, \ldots, \alpha_n$, where $\alpha_n$ is $\alpha$, and each $\alpha_i$ is either an instance of one of the above axioms, a member of $\Gamma$ or obtained by applying modus ponens to previous members of the sequence. We write $\Gamma, \beta \vdash_{\mathrm{PL}} \alpha$ for $(\Gamma \cup \{\beta\}) \vdash_{\mathrm{PL}} \alpha$.

A logic is *sound* if every theorem is valid and it is *complete* if every valid formula is a theorem. PL is both sound and complete [Men87]:

**Theorem 2.2**

$$\vdash_{\mathrm{PL}} \alpha \quad \textit{iff} \quad \models_{\mathrm{PL}} \alpha.$$

When working with extensions of PL such as first order logic (Section 2.2) and modal logics (Chapter 3), the Hilbert axioms of PL are almost never stated/used; all PL tautologies are simply taken for granted. This approach can be justified on the grounds that PL is decidable.

### 2.1.4 Sequent Calculus

In this section we introduce the basic concepts and notions used for a sequent calculus formalism and we define a sequent calculus for PL.

**Definition 2.3** A *sequent* is a pair $(\Gamma, \Delta)$ (written $\Gamma \vdash \Delta$) of finite multisets of formulas. The $\Gamma$ is called the *antecedent* and $\Delta$ the *succedent* of the sequent. If the antecendent $\Gamma$ is empty we write $\vdash \Delta$. If $\Gamma = \{\phi\}$ we write $\phi \vdash \Delta$. Furthermore, we write $\Gamma, \Gamma'$ for $\Gamma \cup \Gamma'$ and $\Gamma, \phi$ for $\Gamma \cup \{\phi\}$. Similar conventions apply for the succedent. If $\Gamma \cap \Delta \neq \emptyset$ then $\Gamma \vdash \Delta$ is called a *basic sequent*.

Note that in this definition, a sequent is a pair of multisets which is also the case in [TS96]. We could instead choose to let a sequent be a pair of sequences [Gen35, Gal86, GLT89] which is the classical way of defining a sequent calculus. We will below say more about multisets versus sequences; as it turns out, there is a quite straightforward and simple relation between the two approaches, and we will actually later work with sequences instead of multisets. Finally, it is also possible to choose ordinary sets as a basis [Pra65, SU98]. This can give rise to some subtle questions, though, and we will not consider it further.

**Definition 2.4** A *sequent rule* is a relation between a (possibly empty) sequence of sequents ($\Gamma_i \vdash \Delta_i$, $1 \leq i \leq n$, $0 \leq n$) and a single sequent ($\Gamma \vdash \Delta$), written

$$\frac{\Gamma_1 \vdash \Delta_1 \quad \Gamma_2 \vdash \Delta_2 \quad \cdots \quad \Gamma_n \vdash \Delta_n}{\Gamma \vdash \Delta} \ .$$

The sequent(s) $\Gamma_i \vdash \Delta_i$ are called the *premise(s)* and $\Gamma \vdash \Delta$ the *conclusion* of the sequent rule.

The *principal formula(s)* of a sequent rule are the formula(s) in the conclusion not occurring in the premise(s). Similarly, the *active formula(s)* are the formula(s) in the premise(s) not occurring in the conclusion.

If a sequent rule has no premise(s) it will alternatively be referred to as an *axiom*.

**Definition 2.5** A finite set of sequent rules $\mathcal{R}$ induces a *sequent calculus* $\mathfrak{G}[\mathcal{R}]$.

A *proof of a sequent* $\Gamma \vdash \Delta$ *in a sequent calculus* $\mathfrak{G}[\mathcal{R}]$ is a finite tree of sequents with $\Gamma \vdash \Delta$ as root. The leaves are either basic sequents or instances of axioms of $\mathcal{R}$. The inner sequents of the tree are connected iff they match an instance of a (non-axiom) sequent rule of $\mathcal{R}$.

Given two sets $\mathcal{R}, \mathcal{R}'$ of sequent rules we write $\mathcal{R}\mathcal{R}'$ for $\mathcal{R} \cup \mathcal{R}'$. Given a sequent rule R we write $\mathcal{R}, \mathrm{R}$ for $\mathcal{R} \cup \{\mathrm{R}\}$.

We say that $\alpha$ is *provable* in, or a *theorem* of, the sequent calculus $\mathfrak{G}[\mathcal{R}]$ (written $\vdash_{\mathfrak{G}[\mathcal{R}]} \alpha$) if there is a proof of the sequent $\vdash \alpha$ in $\mathfrak{G}[\mathcal{R}]$.

After having introduced the basic concepts and notions we will now consider a sequent calculus for PL. Let L denote the set of the following eight sequent rules:

$$\frac{\Gamma, \alpha, \beta \vdash \Delta}{\Gamma, \alpha \wedge \beta \vdash \Delta}\ (\mathrm{L}\wedge) \qquad \frac{\Gamma \vdash \alpha, \Delta \quad \Gamma \vdash \beta, \Delta}{\Gamma \vdash \alpha \wedge \beta, \Delta}\ (\mathrm{R}\wedge)$$

$$\frac{\Gamma, \alpha \vdash \Delta \quad \Gamma, \beta \vdash \Delta}{\Gamma, \alpha \vee \beta \vdash \Delta}\ (\mathrm{L}\vee) \qquad \frac{\Gamma \vdash \alpha, \beta, \Delta}{\Gamma \vdash \alpha \vee \beta, \Delta}\ (\mathrm{R}\vee)$$

$$\frac{\Gamma \vdash \alpha, \Delta \quad \Gamma, \beta \vdash \Delta}{\Gamma, \alpha \to \beta \vdash \Delta}\ (\mathrm{L}\to) \qquad \frac{\Gamma, \alpha \vdash \beta, \Delta}{\Gamma \vdash \alpha \to \beta, \Delta}\ (\mathrm{R}\to)$$

$$\frac{\Gamma \vdash \alpha, \Delta}{\Gamma, \neg\alpha \vdash \Delta}\ (\mathrm{L}\neg) \qquad \frac{\Gamma, \alpha \vdash \Delta}{\Gamma \vdash \neg\alpha, \Delta}\ (\mathrm{R}\neg) \ .$$

The rules are divided in two groups: L(eft) and R(ight) rules, corresponding to whether the principal formula is on the left, respectively the right, side of the turnstile. There is an L and a R rule for each of the connectives $\wedge, \vee, \to, \neg$, which are said to be *introduced* by the corresponding rules.

### Structural Rules

In sequent calculi we must consider certain structural rules, viz.

$$\frac{\Gamma \vdash \Delta}{\Gamma, \alpha \vdash \Delta}\ (\text{LW}) \qquad \frac{\Gamma, \alpha, \alpha \vdash \Delta}{\Gamma, \alpha \vdash \Delta}\ (\text{LC}) \qquad \frac{\Gamma, \beta, \alpha, \Gamma' \vdash \Delta}{\Gamma, \alpha, \beta, \Gamma' \vdash \Delta}\ (\text{LE})$$

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash \alpha, \Delta}\ (\text{RW}) \qquad \frac{\Gamma \vdash \alpha, \alpha, \Delta}{\Gamma \vdash \alpha, \Delta}\ (\text{RC}) \qquad \frac{\Gamma \vdash \Delta, \beta, \alpha, \Delta'}{\Gamma \vdash \Delta, \alpha, \beta, \Delta'}\ (\text{RE})$$

$$\frac{\Gamma \vdash \alpha, \Delta \quad \Gamma, \alpha \vdash \Delta}{\Gamma \vdash \Delta}\ (\text{Cut}) \ .$$

The *exchange rules* ((LE) and (RE)) are obviously superfluous when we consider sequents of multisets; they are on the other hand crucial for sequent calculi based on sequences. It should be clear that a sequent calculus $\mathfrak{G}[\mathcal{R}]$ based on multisets is equivalent to $\mathfrak{G}[\mathcal{R}, (\text{LE}), (\text{RE})]$ based on sequences: Any proof using multisets can be converted to a proof using sequences by just adding a finite number of exchange rules a finite number of places.

Both the *weakening rules* and the *contraction rules* are not necessary for completeness of the version of the sequent calculus for PL we have presented here (see below). We can do without (LW) and (RW) because of the way we have defined a basic sequent (Definition 2.3). We can do without (LC) and (RC) because of the way we have defined the rules of L (in particular, both $\alpha$ and $\beta$ are active formulas in (L$\wedge$) and (R$\vee$)). For more details on these aspects, see, e.g., [Gal86, TS96]. Note that the contraction rules are easily derivable if we include (Cut). We will utilize this fact later.

**Definition 2.6** Given a sequent calculus $\mathfrak{G}[\mathcal{R}, (\text{Cut})]$, we say that *cut-elimination* is possible if for any proof of a sequent $\Gamma \vdash \Delta$ in $\mathfrak{G}[\mathcal{R}, (\text{Cut})]$ there exists a proof of $\Gamma \vdash \Delta$ in $\mathfrak{G}[\mathcal{R}]$. We say that *cut is admissible* for a sequent calculus $\mathfrak{G}[\mathcal{R}]$ if cut elimination is possible for $\mathfrak{G}[\mathcal{R}, (\text{Cut})]$.

We can now state the equivalence with the Hilbert system for PL formally.

**Proposition 2.7**

$$\vdash_{\text{PL}} \alpha \quad \textit{iff} \quad \vdash_{\mathfrak{G}[\text{L}]} \alpha.$$

Proof. The classical proof [Gen35] considers equivalence of the Hilbert system for PL with $\mathfrak{G}[\text{L}, (\text{Cut})]$ and then proves cut-elimination for $\mathfrak{G}[\text{L}, (\text{Cut})]$. When (Cut) is included it is straightforward to prove equivalence as follows: The *if* direction is proved by induction over the length of the proof of $\vdash_{\text{PL}} \alpha$. The *only if* direction is proved by structural induction over the proof tree for $\vdash_{\mathfrak{G}[\text{L},(\text{Cut})]} \alpha$. □

### Structure of Sequent Calculus Rules

The definition of a sequent rule (Definition 2.4) does not specify how the sequents should be related nor the inner structure of the sequents. In [Wan94] some general principles for the structure of the rules for each connective $\circ$ of the logic in question are suggested:

- *Separation.* The sequent rules for $\circ$ should not exhibit any connective other than $\circ$.
- *Weakly symmetric.* The rules for $\circ$ should either be left or right introduction rules.
- *Symmetric.* Both left and right introduction rules for $\circ$.
- *Weakly explicit.* The rules for $\circ$ exhibit $\circ$ only in the conclusion sequents.
- *Explicit.* Only one occurrence of $\circ$ in the conclusion.

A "proper" sequent calculus should ideally satisfy all of these principles. We immediately see that the sequent calculus for PL is proper in this sense.

The five principles are (importantly) not only relevant for "aesthetic" reasons (they make the rules "look nice") but have more profound implications. One is that they make the classical cut-elimination proof for $\mathfrak{G}[\mathrm{L}, (\mathrm{Cut})]$ possible. If some of the principles were not satisfied the proof would become more difficult if not impossible. We will later in this thesis consider a number of logics which do not satisfy all of these principles and thus are less well-behaved.

Another implication is whether the sequent calculus in question satisfies a subformula property.

**Definition 2.8** The *subformula*-relation is the reflexive and transitive closure of: $\alpha$ and $\beta$ are subformulas of $\alpha \wedge \beta$, $\alpha \vee \beta$ and $\alpha \rightarrow \beta$; $\alpha$ is a subformula of $\neg\alpha$. The *proper subformula*-relation is just the transitive closure.

**Definition 2.9** A sequent rule has the *subformula property* if for each active formula $\alpha$ there exists a principal formula $\beta$ such that $\alpha$ is a proper subformula of $\beta$.

If we wish to prove a given sequent in a sequent calculus we can do it by performing a *backwards proof search*. By this we mean that we choose a sequent rule whose conclusion matches the given sequent and we then recursively consider the premises of that rule in turn. If the currently given sequent is a basic sequent, an axiom or no rules have a matching conclusion, we move on to the next sequent; if there are none we stop.

This procedure is of course not guaranteed to terminate in general but in the case of the sequent calculus for PL it is. To show this we associate a measure $\kappa$ with a sequent: $\kappa$ is the sum of the sizes of the formulas of the sequent. It is clear that $\kappa$ of a premise is strictly less than $\kappa$ of the conclusion for any sequent rule satisfying the subformula property.

Now, all sequent rules of L satisfy the subformula property. As a consequence, $\kappa$ will decrease strictly with each step in a backwards proof search in the sequent calculus for PL. This means that a backwards proof search will terminate for any given sequent. We can use this fact to device a decision procedure for PL: Perform a backwards proof search on $\vdash \alpha$; if all leaves in the resulting search tree are either basic sequents or axioms then $\alpha$ is valid, otherwise it is not.

### 2.1.5 Natural Deduction

In this section we consider natural deduction proof systems; in particular a system for PL.

Natural deduction systems are in many respects similar to sequent calculus systems; the most notable being that the ideal "proper" system has two (and only two) rules for each connective and that each such rule concerns that connective only. In a sequent calculus system all assumptions are carried around explicitly whereas they remain implicit in a natural deduction system. This implies a fundamentally different structure to natural deduction rules and we now talk of an *elimination* and an *introduction* rule for each connective (also called E- and I-rules).

These principles are best illustrated by considering a natural deduction system for PL:[4]

$$\frac{\alpha \quad \beta}{\alpha \wedge \beta}\,\wedge I \qquad \frac{\alpha \wedge \beta}{\alpha}\,\wedge E \qquad \frac{\alpha \wedge \beta}{\beta}\,\wedge E$$

$$\frac{\alpha}{\alpha \vee \beta}\,\vee I \qquad \frac{\beta}{\alpha \vee \beta}\,\vee I \qquad \frac{\alpha \vee \beta \quad \begin{array}{c}[\alpha]\\ \vdots\\ \gamma\end{array} \quad \begin{array}{c}[\beta]\\ \vdots\\ \gamma\end{array}}{\gamma}\,\vee E$$

$$\frac{\begin{array}{c}[\alpha]\\ \vdots\\ \beta\end{array}}{\alpha \to \beta}\,\to I \qquad \frac{\alpha \to \beta \quad \alpha}{\beta}\,\to E$$

$$\frac{\begin{array}{c}[\alpha]\\ \vdots\\ \bot\end{array}}{\neg\alpha}\,\neg I \qquad \frac{\alpha \quad \neg\alpha}{\beta}\,\neg E\ .$$

The rules $\vee E$, $\to I$ and $\neg I$ require special attention: The meaning of, say, the rule $\to I$ is that if we by *assuming* $\alpha$ can prove $\beta$ (in some way), then we have proved $\alpha \to \beta$. Similarly for $\vee E$ and $\neg I$. The formulas which can be assumed in rules

[4] We notice that, strictly speaking, we have two E-rules for $\wedge$ and two I-rules for $\vee$. This is just a way of expressing the commutiativity of $\wedge$ and $\vee$ and is thus not essential.

are distinguished by enclosing them in square brackets. These assumed formulas are said to be *closed* when the corresponding rule is used and can then not be used as assumptions for other rules. Assumptions which are not (yet) closed are said to be *open*. As for a sequent rule, if a natural deduction rule has no premise(s) it will be referred to as an *axiom*.

In example derivations, when an assumption is closed by a certain rule, we will use a natural number to identify which occurrence of a rule closes which assumption.

**Definition 2.10** A (natural deduction) *derivation* of a formula $\alpha$ from a set of formulas $\Gamma$ is a proof tree formed using natural deduction rules with $\alpha$ as root and where the leaves are either axioms, closed assumptions or belong to $\Gamma$. If $\Gamma$ is empty, $\alpha$ is said to be a (natural deduction) *theorem*.

We will write $\vdash^{\mathrm{ND}}_{\mathrm{PL}} \alpha$ if $\alpha$ is a theorem of the natural deduction system for PL. The equivalence with the Hilbert system for PL can now be stated formally. The proof is by fairly straightforward (structural) induction.

**Proposition 2.11**

$$\vdash_{\mathrm{PL}} \alpha \quad \textit{iff} \quad \vdash^{\mathrm{ND}}_{\mathrm{PL}} \alpha.$$

In natural deduction there is a notion of a *normal* derivation which (informally speaking) is a derivation with no "detours". The process of *normalization* has many similarities with cut-elimination for sequent calculi. A normal derivation has many nice properties; in particular, it satisfies a *subformula property*.[5] Results concerning these concepts were pioneered by Prawitz in [Pra65]. A contemporary treatment is given in [TS96].

We will not go into a technical discussion of these aspects here but refer to Section 3.5 where a more detailed treatment is given in the context of modal logic.

## 2.2 First Order Logic

### 2.2.1 Syntax

*First Order Logic* (FOL) is an extension of PL where particular structures are imposed on propositional letters. The basic building blocks of FOL are:

1. An infinite set of *variables* $x, y, z, \ldots$.

2. An infinite set of *function symbols* $f^n, g^m, \ldots$ equipped with arities $n, m \geq 0$. If $f^n$ has arity $n = 0$ then $f$ is called a *constant*. Constants will be denoted by $a, b, c, \ldots$.

[5] This is not to be confused with the subformula property of a sequent calculus system.

3. An infinite set of *predicate symbols* $G^n, H^m, \ldots$ equipped with arities $n, m \geq 0$. If $G^n$ has arity $n = 0$ then $G$ corresponds to a propositional letter in PL and will be denoted similarly, i.e., by $p, q, r, \ldots$.

The syntactic category of *terms* $s, t, u, \ldots$ is defined by the following abstract syntax:

$$s \quad ::= \quad x \mid f^n(s_1, \ldots, s_n).$$

In FOL, formulas[6] $\phi, \psi, \chi, \ldots$ are defined by the following abstract syntax:

$$\phi \quad ::= \quad G^n(s_1, \ldots, s_n) \mid \bot \mid \phi \to \psi \mid (\exists x)\phi \mid (\forall x)\phi.$$

Comparing first order formulas with propositional formulas we see that besides the finer structure of propositional letters we have introduced the (standard) existential and universal quantifiers too.[7] As for PL, we can talk of functionally complete sets of operators/quantifiers which means that we can restrict attention to either $\exists$ or $\forall$ as the one is definable in terms of the other.

### 2.2.2 Semantics

We now consider the semantics of FOL. First we need the concept of a model.

**Definition 2.12** A (first order) *model* is a pair $\mathcal{M} = (D, I)$ where $D$ is a non-empty set, the *domain* of $\mathcal{M}$, and $I$ is an *interpretation* mapping each $n$-ary function symbol to an $n$-ary function over $D$, $I(f^n) \in D^n \longrightarrow D$, and mapping each $n$-ary predicate symbol to an $n$-ary relation over $D$, $I(G^n) \subseteq D^n$.

Given a model, we further need a *valuation*[8] $\mathcal{V}$ mapping variables to elements of the domain, $\mathcal{V}(x) \in D$. Two valuations $\mathcal{V}$ and $\mathcal{V}'$ are said to be *$x$-equivalent* iff $\mathcal{V}(y) = \mathcal{V}'(y)$ for any variable $y$ different from $x$.

Given a model and a valuation $\mathcal{V}$ we lift the interpretation $I$ of function symbols to terms by the following inductive definition:

$$\begin{array}{lcl} I_{\mathcal{V}}(x) & = & \mathcal{V}(x), \\ I_{\mathcal{V}}(f^n(s_1, \ldots, s_n)) & = & I(f^n)(I_{\mathcal{V}}(s_1), \ldots, I_{\mathcal{V}}(s_n)). \end{array}$$

Finally, *satisfaction* of a first order formula in a model $\mathcal{M}$ and valuation $\mathcal{V}$ (written $\mathcal{M}, \mathcal{V} \models \phi$) can now be inductively defined as follows:

$$\begin{array}{lll} \mathcal{M}, \mathcal{V} \models G^n(s_1, \ldots, s_n) & \text{iff} & (I_{\mathcal{V}}(s_1), \ldots, I_{\mathcal{V}}(s_n)) \in I(G^n), \\ \mathcal{M}, \mathcal{V} \models \phi \to \psi & \text{iff} & \mathcal{M}, \mathcal{V} \models \phi \text{ implies } \mathcal{M}, \mathcal{V} \models \psi, \\ \mathcal{M}, \mathcal{V} \models (\exists x)\phi & \text{iff} & \mathcal{M}, \mathcal{V}' \models \phi \text{ for some valuation } \mathcal{V}' \\ & & \qquad x\text{-equivalent to } \mathcal{V}. \end{array}$$

[6] To emphasize the distinction we will in this thesis consequently use $\alpha, \beta, \gamma, \ldots$ for propositional formulas and $\phi, \psi, \chi, \ldots$ for first order formulas.

[7] These only make sense in a first order setting, of course.

[8] Note that this is not the same kind of valuation as the one used for PL.

A first order formula $\phi$ is said to be *valid* if $\mathcal{M}, \mathcal{V} \models \phi$ for any model $\mathcal{M}$ and any valuation $\mathcal{V}$.

*First Order Logic with Equality* (FOLE) is FOL where one (binary) predicate symbol is fixed as the equality symbol $=$. Semantically, $=$ is interpreted the obvious way:

$$\mathcal{M}, \mathcal{V} \models s = t \quad \text{iff} \quad I_{\mathcal{V}}(s) = I_{\mathcal{V}}(t).$$

### 2.2.3 Hilbert Systems

First a few technicalities.

**Definition 2.13** A variable $x$ occuring in a formula $\phi$ is said to be *free* if it is not within the scope of $(\forall x)$ or $(\exists x)$, otherwise it is said to be *bound*. By $\mathrm{FV}(\phi)$ we denote the set of free variables of $\phi$. We let $\mathrm{FV}(\Gamma)$ denote the union of the sets of free variables of the formulas in the (multi)set $\Gamma$. A formula with no free variables is said to be *closed*.

In FOL we need the notion of a *substitution* of a term $s$ for a variable $x$ in a formula $\phi$, written $\phi[s/x]$. We adopt Barendregt's variable-convention from the $\lambda$-calculus [Bar84] whereby formulas/terms are identified if they only differ by the names of the bound variables. This implies that a substitution is always possible as bound variables are implicitly renamed to avoid clashes.

A Hilbert proof system for FOL can now be defined as a conservative extension to that for PL with the addition of the following axioms:

Q1: $(\forall x)\phi \to \phi[s/x]$,

Q2: $(\forall x)(\phi \to \psi) \to (\phi \to (\forall x)\psi)$ if $x \notin \mathrm{FV}(\phi)$,

and the *generalization* rule:

G: $\dfrac{\phi}{(\forall x)\phi}$ .

The definition of proof, theorem etc. (Definition 2.1) extends straightforwardly to FOL; the generalization rule is now also applicable (besides MP).

As for PL we have a soundness and completeness result for FOL with respect to the above Hilbert system and the semantics of the previous section. In the case of FOLE this is possible too. We achieve this by adding the following axioms to FOL:

E1: $s = s$,

E2: $s_1 = t_1 \wedge \ldots \wedge s_n = t_n \to f^n(s_1, \ldots, s_n) = f^n(t_1, \ldots, t_n)$,

E3: $s_1 = t_1 \wedge \ldots \wedge s_n = t_n \to (G^n(s_1, \ldots, s_n) \leftrightarrow G^n(t_1, \ldots, t_n))$.

In E3, $G^n$ can in particular be $=$, which means that the well-known axioms saying that $=$ is transitive and symmetric are derivable.

### 2.2.4 Sequent Calculus

Let P denote the set of the following four sequent rules,

$$\frac{\Gamma, \phi[s/x] \vdash \Delta}{\Gamma, (\forall x)\phi \vdash \Delta}\ (\mathrm{L}\forall) \qquad \frac{\Gamma \vdash \phi, \Delta}{\Gamma \vdash (\forall x)\phi, \Delta}\ (\mathrm{R}\forall)$$

$$\frac{\Gamma, \phi \vdash \Delta}{\Gamma, (\exists x)\phi \vdash \Delta}\ (\mathrm{L}\exists) \qquad \frac{\Gamma \vdash \phi[s/x], \Delta}{\Gamma \vdash (\exists x)\phi, \Delta}\ (\mathrm{R}\exists)\ ,$$

where the following sideconditions apply:

- (L$\exists$) and (R$\forall$): $x \notin \mathrm{FV}(\Gamma \cup \Delta)$.

As for the sequent calculus for PL, the weakening rules are not necessary for completeness. But contrary to PL, we do need the contraction rules to secure completeness for FOL. In other words, $\mathfrak{G}[\mathrm{LP}, (\mathrm{LC}), (\mathrm{RC})]$ gives us FOL. It is however possible to absorb the contraction rules in the quantifier rules of P [TS96]. This is more specifically done by duplicating the principal formula in the premise beside the active formula in the rules (L$\forall$) and (R$\exists$):

$$\frac{\Gamma, \phi[s/x], (\forall x)\phi \vdash \Delta}{\Gamma, (\forall x)\phi \vdash \Delta}\ (\mathrm{L}\forall') \qquad \frac{\Gamma \vdash \phi[s/x], (\exists x)\phi, \Delta}{\Gamma \vdash (\exists x)\phi, \Delta}\ (\mathrm{R}\exists')\ .$$

With these rules we have soundness and completeness for FOL without any of the structural rules.

We extend Definition 2.8 to FOL by saying that $\phi[s/x]$ is a subformula of $(\forall x)\phi$ and $(\exists x)\phi$ for any $s$. This means that the rules of P satisfy the subformula property whereas the rules (L$\forall'$) and (R$\exists'$) do not. Hence, in the latter case we do not have a decision procedure for FOL as we do for PL. But neither do we in the former case since we there had to include the contraction rules which clearly do not satisfy the subformula property themselves. In fact, FOL is provably undecidable.

We now turn our attention to FOLE. Let E denote the set of the following sequent rules ($1 \leq i \leq n$):[9]

$$\frac{\Gamma, s = s \vdash \Delta}{\Gamma \vdash \Delta}\ (\mathrm{E1})$$

$$\frac{\Gamma \vdash s_i = t_i, \Delta \quad \Gamma, f^n(s_1, \ldots, s_n) = f^n(t_1, \ldots, t_n) \vdash \Delta}{\Gamma \vdash \Delta}\ (\mathrm{E2})$$

[9] The form of these rules is inspired by [Gal86, pp. 236–237]; they are slightly modified such as to make some proofs in Section 5.2.2 simpler.

$$\frac{\Gamma \vdash s_i = t_i, \Delta \quad \Gamma \vdash G^n(s_1, \ldots, s_n), \Delta \quad \Gamma, G^n(t_1, \ldots, t_n) \vdash \Delta}{\Gamma \vdash \Delta}\ (\text{E3}) \ .$$

By adding these rules to the sequent calculus for FOL we achieve equivalence with FOLE.

To ease later references we note the following: If the only predicate symbol is $=$, (E3) corresponds to:

$$\frac{\Gamma \vdash s_1 = t_1, \Delta \quad \Gamma \vdash s_2 = t_2, \Delta \quad \Gamma \vdash s_1 = s_2, \Delta \quad \Gamma, t_1 = t_2 \vdash \Delta}{\Gamma \vdash \Delta}\ (\text{EE}) \ ,$$

from which the following two sequent rules are easily derivable.

$$\frac{\Gamma \vdash s = t, \Delta \quad \Gamma \vdash t = u, \Delta \quad \Gamma, s = u \vdash \Delta}{\Gamma \vdash \Delta}\ (\text{ET})$$

$$\frac{\Gamma \vdash s = t, \Delta \quad \Gamma, t = s \vdash \Delta}{\Gamma \vdash \Delta}\ (\text{ES}) \ .$$

### 2.2.5 Natural Deduction

Finally, in this last section, we consider natural deduction systems for FOL and FOLE.

We get a system for FOL by extending the system for PL with the following rules:

$$\frac{\phi}{(\forall x)\phi}\ \forall I \qquad \frac{(\forall x)\phi}{\phi[s/x]}\ \forall E$$

$$\frac{\phi[s/x]}{(\exists x)\phi}\ \exists I \qquad \frac{(\exists x)\phi \quad \begin{array}{c}[\phi]\\ \vdots \\ \psi\end{array}}{\psi}\ \exists E \ ,$$

where the following sideconditions apply:

- $\forall I$: $x$ is not free in any assumption on which $\phi$ depends.
- $\exists E$: $x$ is not free in $\psi$ nor in any assumption on which $\psi$ depends except $\phi$.

If we further extend the system for FOL with the following rules:

$$\frac{\phi[s/x] \quad s = t}{\phi[t/x]}\ \text{Subst} \qquad \frac{}{s = s}\ \text{Refl} \ ,$$

we have a system for FOLE.

CHAPTER 3

# Modal Logic

This chapter discusses modal logic, with emphasis on proof theory. We consider both propositional and first order variants, the classical modal logics with two unary modalities and modal logics with a binary modality.

Some good references for propositional modal logics are [HC68, Che80, BS84, HC84]. First order modal logics are discussed in [Gar84, HC96, FM98]. On the proof theory of modal logics specifically we refer to [Fit83, Vig00].

## 3.1 Syntax

Syntactically, modal logics are simple extensions of PL/FOL with the introduction of one or more *modalities*. The best-known examples of such modalities are the unary $\Box$ and $\Diamond$. Thus, formulas can have the forms $\Box\alpha$ and $\Diamond\alpha$ as well in those cases.

Syntactically, (Boolean) operators and modalities are not distinguished: We collectively refer to operators, quantifiers and modalities as *connectives*. The definition of an atomic formula is accordingly extended to formulas with no connectives. Similarly, the size of a formula is now given by the number of connectives in the formula.

As for PL and FOL we can restrict attention to functionally complete sets of connectives[1] for modal logics. In the case of the classical logics with $\Box$ and $\Diamond$ we can take either as basic and the other can then be defined in terms of the first. Taking $\Box$ as basic (in which case $\Diamond$ can be defined as $\Diamond\alpha \mathrel{\widehat{=}} \neg(\Box(\neg\alpha))$) we will collectively refer to these modal logics by $\mathcal{L}^{\Box}$.

[1] We now include modalities in these sets as well.

Modalities are not necessarily unary; a very important role in this thesis is played by the binary *chop*: $\frown$ (i.e., we have formulas $\alpha\frown\beta$ besides the usual PL/FOL formulas). We will collectively refer to such logics (with only the binary chop modality) by $\mathcal{L}^{\frown}$.

Finally, modalities will have precedence over all Boolean operators except $\neg$ which has precedence over modalities.

## 3.2 Semantics

Modal logics are traditionally given a so-called Kripke-style semantics (also called possible worlds semantics).

**Definition 3.1** A (propositional modal logic) *model* is a triple $\mathfrak{M} = (W, R, \mathcal{U})$, where $W$ is a non-empty set of *(possible) worlds*, $R$ is an *(accessibility) relation* over $W$, and $\mathcal{U}$ is a valuation mapping propositional letters to subsets of $W$ ($\mathcal{U}$ corresponds to the valuation function of propositional logic but is now, importantly, dependent on the worlds[2]).

The arity of the relation $R$ is dependent on the arity of the modalities of the logic in question. For $\mathcal{L}^{\Box}$ it is binary, for $\mathcal{L}^{\frown}$ it is ternary. The pair $(W, R)$ is called the *frame* of the model and we say that a model $(W, R, \mathcal{U})$ is *based on* the frame $(W, R)$.

We define satisfaction of a formula $\alpha$ in a world $w \in W$ in a model $\mathfrak{M} = (W, R, \mathcal{U})$ (written $\mathfrak{M}, w \models \alpha$) as follows:

$$\begin{array}{lll} \mathfrak{M}, w \models p & \text{iff} & w \in \mathcal{U}(p), \\ \mathfrak{M}, w \models \alpha \to \beta & \text{iff} & \mathfrak{M}, w \models \alpha \text{ implies } \mathfrak{M}, w \models \beta. \end{array}$$

Here we have given the semantics of propositional letters and the Boolean operators. The interpretation of propositional letters is dependent on the world whereas the Boolean operators are independent. We now consider the semantics for the more interesting cases concerning the modalities. For $\mathcal{L}^{\Box}$ we have:

$$\mathfrak{M}, w \models \Box\alpha \quad \text{iff} \quad \mathfrak{M}, v \models \alpha \text{ for all } v \in W \text{ where } R(v, w).$$

In the case of $\mathcal{L}^{\frown}$ we have:

$$\begin{array}{lll} \mathfrak{M}, w \models \alpha\frown\beta & \text{iff} & \mathfrak{M}, v \models \alpha \text{ and } \mathfrak{M}, u \models \beta \text{ and} \\ & & R(v, u, w) \text{ for some } v, u \in W. \end{array}$$

We say that $\alpha$ is *satisfiable in a class of models* $\mathcal{C}$ if there is a model $\mathfrak{M}$ of $\mathcal{C}$ and a world $w$ of $\mathfrak{M}$ such that $\mathfrak{M}, w \models \alpha$. Given a set of formulas $\Gamma$, we say that $\mathfrak{M}$ satisfies $\Gamma$ if there is a world $w$ of $\mathfrak{M}$ such that $\mathfrak{M}, w \models \alpha$ for every $\alpha \in \Gamma$.

A formula $\alpha$ is *valid in a model* $\mathfrak{M}$ if for all worlds $w$ of $\mathfrak{M}$, $\mathfrak{M}, w \models \alpha$. A formula $\alpha$ is *valid in a class of models* $\mathcal{C}$ if it is valid in all models of $\mathcal{C}$. Furthermore, a

[2] $\mathcal{U}(\bot) = \emptyset$ for any $\mathcal{U}$.

formula $\alpha$ is *valid in a frame* $F$ if it is valid in all models based on $F$. Finally, a formula $\alpha$ is *valid in a class of frames* $\mathfrak{F}$ if it is valid in all frames of $\mathfrak{F}$.

We have so far not said anything about possible restrictions on $R$. We will in the next section see how the well-known properties of reflexitivity, symmetry, etc. have nice connections in the proof systems for $\mathcal{L}^{\Box}$ logics.

We now turn to the question of semantics for first order modal logics. For this we consider the amalgamation of the two kinds of models of Definition 2.12 and Definition 3.1. But to do this we have to address some points. First, is the domain $D$ the same in all worlds? There has been much work on this topic from both a mathematical and a philosophical viewpoint. Some good reference are, e.g., [Gar84, HC96, FM98]. For the purpose of this thesis we will only consider models with constant domains, i.e., the domain is the same in all worlds. Another question (in some sense orthogonal) is whether the same symbol can be interpreted differently in different worlds? The possibility of this will be important in this thesis. We thus for our purposes define a model as follows:

**Definition 3.2** A (first order modal logic with constant domain) *model* is a quadruple $\mathfrak{M} = (W, R, D, I)$ where $W$ and $R$ are as in Definition 3.1, $D$ and $I$ are as in Definition 2.12 with the important modification that $I$ now also depends on the worlds, thus $I(f^n)(w) \in D^n \longrightarrow D$ and $I(G^n)(w) \subseteq D^n$.

One could also ask whether the valuation (potentially) depends on the worlds. We will not consider this, i.e., we consider only valuations as the ones in Section 2.2.2.

The definition of the first order semantics of terms and formulas is a straightforward amalgamation of the definitions in Section 2.2.2 and above.

The interpretation $I$ is lifted to terms:

$$\begin{array}{lcl} I_{\mathcal{V}}(x)(w) & = & \mathcal{V}(x), \\ I_{\mathcal{V}}(f^n(s_1, \ldots, s_n))(w) & = & I(f^n)(w)(I_{\mathcal{V}}(s_1)(w), \ldots, I_{\mathcal{V}}(s_n)(w)), \end{array}$$

and the satisfaction of first order modal formulas is defined as follows:

$$\begin{array}{lll} \mathfrak{M}, \mathcal{V}, w \models G^n(s_1, \ldots, s_n) & \text{iff} & (I_{\mathcal{V}}(s_1)(w), \ldots, I_{\mathcal{V}}(s_n)(w)) \in I(G^n)(w), \\ \mathfrak{M}, \mathcal{V}, w \models \phi \rightarrow \psi & \text{iff} & \mathfrak{M}, \mathcal{V}, w \models \phi \text{ implies } \mathfrak{M}, \mathcal{V}, w \models \psi, \\ \mathfrak{M}, \mathcal{V}, w \models (\exists x)\phi & \text{iff} & \mathfrak{M}, \mathcal{V}', w \models \phi \text{ for some valuation } \mathcal{V}' \\ & & \qquad x\text{-equivalent to } \mathcal{V}, \\ \mathfrak{M}, \mathcal{V}, w \models \Box\alpha & \text{iff} & \mathfrak{M}, \mathcal{V}, v \models \alpha \text{ for all } v \in W \text{ where } R(v, w), \\ \mathfrak{M}, \mathcal{V}, w \models \alpha^\frown\beta & \text{iff} & \mathfrak{M}, \mathcal{V}, v \models \alpha \text{ and } \mathfrak{M}, \mathcal{V}, u \models \beta \text{ and} \\ & & \qquad R(v, u, w) \text{ for some } v, u \in W. \end{array}$$

## 3.3 Hilbert Systems

We now turn to Hilbert proof systems for modal logics. We first consider some examples of propositional $\mathcal{L}^{\Box}$ logics.

The simplest of these is the modal logic K which is an extension of PL with a rule of necessitation

$$\frac{\alpha}{\Box\alpha},$$

and the following axiom:

$$\Box(\alpha \to \beta) \to (\Box\alpha \to \Box\beta).$$

The logic K is sound and complete with respect to the (propositional) semantics of the previous section where, in particular, no restrictions have been put on the accessibility relation $R$. If we require $R$ to be reflexive we have soundness and completeness with respect to the logic T, which is K with the addition of the axiom

$$\Box\alpha \to \alpha.$$

In a similar way, by extending T with the axiom

$$\Box\alpha \to \Box\Box\alpha,$$

we get the modal logic S4 for which soundness and completeness is achieved by requiring $R$ to be both reflexive and transitive. Finally, if $R$ is an equivalence relation the corresponding modal logic is S5, which is S4 with the additional axiom

$$\alpha \to \Box\Diamond\alpha.$$

Similar and more advanced (so called) correspondence results are discussed in [vB84].

The above correspondence results can be "lifted" to first order modal logics as follows: Simply add the first order axioms Q1 and Q2, as well as the generalization rule G, to the propositional modal logic in question. By furthermore adding the *Barcan formula*,

$$(\forall x)\Box\phi \to \Box(\forall x)\phi,$$

we achieve soundness and completeness with respect to the constant domain semantics (Definition 3.2) with the same restrictions on the accessibility relation $R$ as in the propositional cases. For what happens in systems with non-constant domains, Hilbert systems without the Barcan formula etc., we refer to, e.g., [Gar84, HC84, HC96, FM98].

We now turn to discuss Hilbert proof systems for propositional $\mathcal{L}^{\frown}$ logics. A *logic with a binary modality* [KNSS95, AKN$^+$96, MV97] is a $\mathcal{L}^{\frown}$ logic (being an extension of PL) which includes axioms saying that $\frown$ distributes over $\vee$:

$$\text{K:} \qquad \begin{array}{lcl} \alpha\frown(\beta \vee \gamma) & \to & (\alpha\frown\beta) \vee (\alpha\frown\gamma), \\ (\alpha \vee \beta)\frown\gamma & \to & (\alpha\frown\gamma) \vee (\beta\frown\gamma), \end{array}$$

and the following monotonicity rules:

$$\text{M:} \qquad \frac{\alpha \to \beta}{(\alpha \frown \gamma) \to (\beta \frown \gamma)} \qquad \frac{\alpha \to \beta}{(\gamma \frown \alpha) \to (\gamma \frown \beta)} \quad .$$

The *minimal* logic with a binary modality is the logic with a binary modality consisting only of the above axioms and rules. If we include necessitation rules:

$$\text{N:} \qquad \frac{\alpha}{\neg(\neg\alpha \frown \beta)} \qquad \frac{\alpha}{\neg(\beta \frown \neg\alpha)} \quad ,$$

we call the logic *normal*. We can thus speak of the minimal normal logic with a binary modality. We will denote this specific logic $\mathcal{L}^{\frown}_{\mathrm{AF}}$ and write $\vdash_{\mathrm{AF}} \alpha$ if $\alpha$ is a theorem of $\mathcal{L}^{\frown}_{\mathrm{AF}}$.

The "AF" in $\mathcal{L}^{\frown}_{\mathrm{AF}}$ is short for "All Frames", the reason being that if we let $\models_{\mathrm{AF}} \alpha$ denote validity of $\alpha$ in the class of all frames we have the following soundness and completeness result [KNSS95, MV97]:

**Theorem 3.3**

$$\models_{\mathrm{AF}} \alpha \quad \textit{iff} \quad \vdash_{\mathrm{AF}} \alpha.$$

Consider an associativity axiom for $\frown$:

$$\text{A:} \qquad \alpha \frown (\beta \frown \gamma) \;\leftrightarrow\; (\alpha \frown \beta) \frown \gamma.$$

We can now speak of an *associative* $\mathcal{L}^{\frown}$ logic, and thus of the minimal associative (normal) logic with a binary modality.

As for the $\mathcal{L}^{\Box}$ logics above we will consider what happens when a certain structure is imposed on $R$ [KNSS95, AKN$^+$96, MV97].

**Definition 3.4** Given a non-empty set $T$, a *square frame* $(W, R)$ is defined as follows:

- $W = T \times T$,
- $R = \{((i,k),(k,j),(i,j)) \mid i,j,k \in T\}$.

A *square model* is a model based on a square frame.

Notice how the definition of the semantics of $\frown$ can be simplified if we assume a square model:

$$\mathfrak{M}, (i,j) \models \alpha \frown \beta \qquad \text{iff} \qquad \mathfrak{M}, (i,k) \models \alpha \text{ and } \mathfrak{M}, (k,j) \models \beta \text{ for some } k \in T.$$

We are interested in validity of a formula $\alpha$ in the class of all square frames, written $\models_{\mathrm{SQ}} \alpha$. A logic is called a *square extension* of the minimal logic with a

binary modality if all theorems of the logic are valid in the class of all square frames. It is easy to check that the associativity axiom is valid in the class of all square frames. We can thus speak of a square extension of the minimal associative logic with a binary modality.

We now consider the logic $\mathcal{L}^{\frown}_{\mathrm{SQ}}$ with a binary modality characterized by the class of all square frames. By this we mean the logic whose theorems are exactly those valid in all square frames. This logic is a square extension of the minimal associative logic by the above definitions. It would be nice if $\mathcal{L}^{\frown}_{\mathrm{SQ}}$ was simply the minimal associative [normal] logic with a binary modality. Unfortunately, $\mathcal{L}^{\frown}_{\mathrm{SQ}}$ is not even finitely axiomatizable [MV97].

We now cite a central result of [KNSS95].

**Theorem 3.5** *Any square extension of the minimal associative logic with a binary modality is undecidable.*

Hence, it is undecidable whether $\models_{\mathrm{SQ}} \alpha$.

We will not discuss Hilbert systems for first order $\mathcal{L}^{\frown}$ logics in general here. In Chapter 4 we will more thoroughly consider certain extensions to first order $\mathcal{L}^{\frown}$ logics in connection with interval logics.

## 3.4 Sequent Calculus

In this section we consider sequent calculus proof systems for modal logics. This turns out to be considerably more difficult than giving Hilbert proof systems. The nice correspondences of the Hilbert systems are not apparent in the sequent calculi. This is in fact a major problem for modal logics in general [BS84].

We will here concentrate on the modal logic S4 as this logic can be given a reasonably nice sequent calculus formulation. It will furthermore be relevant for us later in the thesis.

What seems to be the first formulation of a sequent calculus for S4, with a proof of cut-elimination, is [Cur52]. We will in the following consider the standard (contemporary) presentation for S4, see, e.g., [TS96].

**Definition 3.6** Let $\Gamma = \{\phi_1, \phi_2, \ldots, \phi_n\}$ be a (finite) multiset of formulas. Then

$$\begin{aligned} \Box\Gamma &\mathrel{\widehat{=}} \{\Box\phi_1, \Box\phi_2, \ldots, \Box\phi_n\}, \\ \Diamond\Gamma &\mathrel{\widehat{=}} \{\Diamond\phi_1, \Diamond\phi_2, \ldots, \Diamond\phi_n\}. \end{aligned}$$

Sequent rules for S4 can now be stated as follows:

$$\frac{\Gamma, \phi \vdash \Delta}{\Gamma, \Box\phi \vdash \Delta}\ (\mathrm{L}\Box) \qquad \frac{\Box\Gamma \vdash \phi, \Diamond\Delta}{\Box\Gamma \vdash \Box\phi, \Diamond\Delta}\ (\mathrm{R}\Box)$$

$$\frac{\Box\Gamma, \phi \vdash \Diamond\Delta}{\Box\Gamma, \Diamond\phi \vdash \Diamond\Delta}\ (\mathrm{L}\Diamond) \qquad \frac{\Gamma \vdash \phi, \Delta}{\Gamma \vdash \Diamond\phi, \Delta}\ (\mathrm{R}\Diamond)\ .$$

Notice that sideconditions are "hidden" in the rules (R $\Box$) and (L$\Diamond$) by means of Definition 3.6. These two rules can informally be read as follows: If $\Gamma$ and $\Delta$ only contain formulas of the form $\Box\phi$ and $\Diamond\phi$, respectively, then the rules can be applied.

Definition 2.8 is straightforwardly extended to cover any additional connectives we introduce — including $\Box$ and $\Diamond$. Hence, we conclude that the above four rules satisfy the subformula property.

As for FOL, we do not have soundness and completeness without some of the structural rules. In this case we actually need both the contraction and the weakening rules. In the case of contraction we can absorb them in the rules (as for FOL) by duplicating the principal formula in the premise beside the active formula in the rules (L$\Box$) and (R$\Diamond$):

$$\frac{\Gamma, \Box\phi, \phi \vdash \Delta}{\Gamma, \Box\phi \vdash \Delta}\ (\mathrm{L}\Box') \qquad \frac{\Gamma \vdash \phi, \Diamond\phi, \Delta}{\Gamma \vdash \Diamond\phi, \Delta}\ (\mathrm{R}\Diamond') \ .$$

The weakening rules also turn out absorbable [TS96]: This is done by modifying the sequent rules (L$\Diamond$) and (R$\Box$) as follows:

$$\frac{\Box\Gamma, \phi \vdash \Diamond\Delta}{\Gamma', \Box\Gamma, \Diamond\phi \vdash \Diamond\Delta, \Delta'}\ (\mathrm{L}\Diamond'') \qquad \frac{\Box\Gamma \vdash \phi, \Diamond\Delta}{\Gamma', \Box\Gamma \vdash \Box\phi, \Diamond\Delta, \Delta'}\ (\mathrm{R}\Box'') \ .$$

In Chapter 5 we will consider a particular formulation for S4, namely that with weakening but not contraction absorbed. In other words, we will consider the rules (L$\Box$), (R$\Box''$), (L$\Diamond''$) and (R$\Diamond$) which together will be denoted "4".

### 3.4.1 Sequent Calculi for Modal Logics

We saw above how S4 can be given a "reasonably nice" sequent calculus formulation. Unfortunately, this is more the exception than the rule when we consider modal logics in general — even when we just look at other simple modal logics such as K, T and S5 [BS84]. In particular, a major problem is the difficulty of defining a common framework in which the modal logics can be presented in a modular way — corresponding to the nice way the logics were defined as simple conservative extensions of each other in a Hilbert system.

Various proposals for sequent calculus systems for some of the simple modal logics (K,T,S4,S5) are surveyed in the introduction of [Wan94]. It is interesting to note that few of these systems satisfy all of the principles discussed in Section 2.1.4.

In [Gor92] cut-free systems for other modal logics (known as S4.3, S4.3.1, and S4.14) are given. It is still within the standard sequent calculus framework but now the rules themselves get more complicated, such that, e.g., the subformula property does not hold any more. The rules are still *analytic* in the sense that if the conclusion is known then the premise(s) are completely determined.

In [Ser82] the author gives cut-free systems for T, S4 and S5 (and the systems known as S4.2 and Br) by putting various restrictions on the rules (L $\Box$) and (R$\Box$).

(only formulations for S4.2 are not seen before). These sideconditions can get rather complicated.

As a way of overcoming these problems, there have been various proposals for generalized systems based on extended formalisms (in contrast to standard sequent calculi systems). This way it is possible to get rid of many of the special sideconditions etc., and thereby have a clean presentation of many modal logics in a common framework. The downside is that the systems themselves (used to present the logics) get more complicated. Examples include [Doŝ85] (two-level sequent calculus), [Mas92] (2-Sequent Calculus), [Cer93] (signed modal sequents) and [Wan94] (display logic [Bel82]). The latter reference contains a survey of the work of the former three.

## 3.5 Natural Deduction

In this section we consider natural deduction systems for modal logics. The classical systems essentially face the same difficulties as the sequent calculus systems of the previous section. This is, e.g., the case for [Pra65, BM92]. In fact, all classical proof theoretical formalisms (with, of course, the notable exception of Hilbert systems) do not seem suitable for presenting modal logics. A formalism which was mentioned in Section 1.2 was that of semantic tableaux. In [Fit83] this formalism is extended to cope with modal logics but the inherent difficulties are still apparent. The ideas are further exploited in [Wal90, Gor92].

We will here not say more about the above work but instead turn to a formalism known as *Labelled Natural Deduction* (LND) [Vig00]. This proof theoretical framework is fundamentally different from the approaches considered so far: The possible worlds of the modal logics, which so far only have appeared in the semantics, are made *part of* the syntax and thereby an important part of the proof system. This approach is taken in [Vig00] which describes work carried out in response to Gabbay's program on labelled deductive systems [Gab96] initiated in the beginning of the 1990's.

The most important consequence of the LND formalism is that it is possible to have a "proper" natural deduction system with exactly one I- and one E-rule for each connective — including the modalities.

We refer to [Vig00] for a thorough treatment of how LND systems can be defined for the classical propositional modal logics (K, T, S4, S5), other variants and first order variants as well — all in a common framework. Below we will concentrate on giving a detailed presentation of a LND system for logics with a binary modality. This will act as a basis for later developments in the thesis.

### 3.5.1 Labelled Natural Deduction for Logics with a Binary Modality

**Definition 3.7** A *labelled formula* is a pair of a possible world $w \in W$ and a formula $\alpha$, written $w : \alpha$. A *relational formula* is a triple of possible worlds $v, u, w$, written $\mathcal{R}(v, u, w)$. We let $\eta$ denote an arbitrary labelled/relational formula.

We define satisfaction of $\eta$ in a model $\mathfrak{M} = (W, R, \mathcal{U})$ (written $\mathfrak{M} \Vdash \eta$) as follows:

| | | |
|---|---|---|
| $\mathfrak{M} \Vdash \mathcal{R}(v,u,w)$ | iff | $R(v,u,w)$, |
| $\mathfrak{M} \Vdash w : p$ | iff | $w \in \mathcal{U}(p)$, |
| $\mathfrak{M} \Vdash w : \alpha \to \beta$ | iff | $\mathfrak{M} \Vdash w : \alpha$ implies $\mathfrak{M} \Vdash w : \beta$, |
| $\mathfrak{M} \Vdash w : \alpha^\frown\beta$ | iff | $\mathfrak{M} \Vdash v : \alpha$, $\mathfrak{M} \Vdash u : \beta$ and $\mathfrak{M} \Vdash \mathcal{R}(v,u,w)$ for some $v, u \in W$. |

We say that a labelled formula $w : \alpha$ is valid in a class of frames $\mathfrak{F}$ if for all frames $F$ of $\mathfrak{F}$, for all models $\mathfrak{M}$ based on $F$, $\mathfrak{M} \Vdash w : \alpha$.

It is clear that $\mathfrak{M} \Vdash w : \alpha$ iff $\mathfrak{M}, w \models \alpha$. Thus, if we let $\Vdash_{\mathrm{AF}} w : \alpha$ denote validity of $w : \alpha$ in the class of all frames we have

**Proposition 3.8**

$$\models_{\mathrm{AF}} \alpha \quad \textit{iff} \quad \Vdash_{\mathrm{AF}} w : \alpha \ \textit{ for all } w \in W.$$

We now define a LND system for $\mathcal{L}^{\frown}_{\mathrm{AF}}$ (inspired by [BMV97, BMV98b]):

$$\frac{\begin{array}{c}[w:\alpha]\\ \vdots \\ w:\beta\end{array}}{w:\alpha\to\beta}\to I \qquad \frac{w:\alpha\to\beta \quad w:\alpha}{w:\beta}\to E \qquad \frac{\begin{array}{c}[w:\alpha\to\bot]\\ \vdots \\ v:\bot\end{array}}{w:\alpha}\bot E$$

$$\frac{v:\alpha \quad u:\beta \quad \mathcal{R}(v,u,w)}{w:\alpha^\frown\beta}\frown I \qquad \frac{w:\alpha^\frown\beta \qquad \begin{array}{c}[v:\alpha]\ [u:\beta]\ [\mathcal{R}(v,u,w)]\\ \vdots \\ w':\gamma\end{array}}{w':\gamma}\frown E \quad ,$$

with the following sidecondition:

- $\frown E$: $v$ and $u$ are different from both $w, w'$ and each other, and do not occur in any assumption on which the upper occurrence of $w' : \gamma$ depends except $v : \alpha, u : \beta$ and $\mathcal{R}(v,u,w)$.

We have here only given LND rules for a functionally complete set of connectives. Rules for the remaining connectives (in the style of the rules in Section 2.1.5) could easily be derived.

The rule $\bot E$ can be regarded as an E-rule for $\bot$ (hence the name) but when we henceforth collectively refer to the E-rules of the system this will *not* include $\bot E$.

We extend Definition 2.10 to an LND system:

**Definition 3.9** A (LND) *derivation* of a labelled formula $w : \alpha$ from a set of labelled formulas $\Gamma$ and a set of relational formulas $\Delta$ is a proof tree formed using LND rules with $w : \alpha$ as root and where the leaves are either axioms, closed assumptions or belong to $\Gamma$ or $\Delta$. We will use $\Pi$ to denote such a derivation. If $\Gamma$ and $\Delta$ are empty, we say that $w : \alpha$ is a (LND) *theorem*.

We write $\vdash^{\mathrm{LND}}_{\mathrm{AF}} w : \alpha$ if $w : \alpha$ is a theorem in the LND system for $\mathcal{L}^{\frown}_{\mathrm{AF}}$.

In [BMV98b] a general soundness and completeness result for labelled propositional logics with $n$-ary modalities is proved. We can utilize this to get:

**Theorem 3.10**

$$\Vdash_{\mathrm{AF}} w : \alpha \quad \textit{iff} \quad \vdash^{\mathrm{LND}}_{\mathrm{AF}} w : \alpha.$$

The soundness and completeness result of [BMV98b] also covers the more general cases where the properties of the relation $R$ are specified within the natural deduction system as Horn clauses relating $\mathcal{R}$ judgments. We will not consider this here.

If we are only interested in completeness with respect to the standard semantics we can prove it more directly:

**Proposition 3.11**

$$\models_{\mathrm{AF}} \alpha \quad \textit{iff} \quad \vdash^{\mathrm{LND}}_{\mathrm{AF}} w : \alpha \textit{ for all } w \in W.$$

Proof. Soundness is straightforward; using Proposition 3.8 we show that the LND rules preserve labelled validity.[3] For the completeness part we utilize Theorem 3.3: We show that if $\vdash_{\mathrm{AF}} \alpha$ then $\vdash^{\mathrm{LND}}_{\mathrm{AF}} w : \alpha$ for all $w$. This can be shown by induction on the length of the proof of $\vdash_{\mathrm{AF}} \alpha$ which amounts to showing that all axioms are provable and that the rules preserve provability in the LND system. The propositional part is standard, hence we restrict our attention to K, M and N. We here only show the latter case:

$$\dfrac{\dfrac{[w : (\alpha \to \bot)\frown\beta]^1 \qquad \dfrac{\dfrac{[v : \alpha \to \bot]^2 \quad v : \alpha}{v : \bot}\to E}{w : \bot}\bot E}{w : \bot}\frown E^2}{w : (\alpha \to \bot)\frown\beta \to \bot}\to I^1 \quad .$$

This derivation is valid as $\vdash^{\mathrm{LND}}_{\mathrm{AF}} u : \alpha$ for all $u$ by the induction hypothesis. We can in particular assume $\vdash^{\mathrm{LND}}_{\mathrm{AF}} v : \alpha$. □

### Normalization

We now turn to consider *normalization* properties for the LND system for $\mathcal{L}^{\frown}_{\mathrm{AF}}$. To be able to establish normalization results for a natural deduction system means that we can very precisely characterize the structure of derivations.

From the semantics we observe that $\frown$ is an existential $\Diamond$-like binary modality. The normalization proofs get simpler if we instead consider an universal $\Box$-like binary modality $\smile$. We define $\smile$ by the following LND rules:

[3] This is (not surprisingly) also the way soundness of Theorem 3.10 is proved in [BMV98b].

$$\dfrac{\begin{array}{c}[v:\alpha][\mathcal{R}(v,u,w)]\\ \vdots\\ u:\beta\end{array}}{w:\alpha\breve{\frown}\beta}\breve{\frown}I \qquad \dfrac{w:\alpha\breve{\frown}\beta \quad v:\alpha \quad \mathcal{R}(v,u,w)}{u:\beta}\breve{\frown}E\ ,$$

with the following sidecondition:

- $\breve{\frown}I$: $v$ and $u$ are different from both $w$ and each other, and $u$ does not occur in any assumption on which $u:\beta$ depends other than $\mathcal{R}(v,u,w)$.

We can show that $\{\bot,\rightarrow,\breve{\frown}\}$ is a functionally complete set of connectives for $\mathcal{L}_{\widehat{\mathrm{AF}}}$ by defining $w:\alpha\frown\beta \mathrel{\widehat{=}} w:\neg(\alpha\breve{\frown}\neg\beta)$ and showing that the rules for $\frown$ can be derived from those for $\breve{\frown}$. Below we give the case for $\frown I$ (remember that $\neg\alpha \mathrel{\widehat{=}} \alpha\rightarrow\bot$):

$$\dfrac{\dfrac{\dfrac{[w:\alpha\breve{\frown}(\beta\rightarrow\bot)]^1 \quad v:\alpha \quad \mathcal{R}(v,u,w)}{u:\beta\rightarrow\bot}\breve{\frown}E \qquad u:\beta}{\dfrac{u:\bot}{w:\bot}\bot E}\rightarrow E}{w:\alpha\breve{\frown}(\beta\rightarrow\bot)\rightarrow\bot}\rightarrow I^1 \ .$$

We could similarly derive the rules for $\breve{\frown}$ from those for $\frown$. In the following we will restrict attention to the $\{\bot,\rightarrow,\breve{\frown}\}$ fragment.

**Proposition 3.12** *For any derivation of $w:\alpha$ in the LND system for $\mathcal{L}_{\widehat{\mathrm{AF}}}$ there is a derivation of $w:\alpha$ with the following restrictions on the $\bot E$ rule: 1) The conclusion is always atomic, and 2) there are no applications immediately following each other.*

Proof. 1) In the original derivation, pick out an application of $\bot E$ where the conclusion has maximal size. If not atomic (in which case we are done), this conclusion will have form $\alpha\rightarrow\beta$ or $\alpha\breve{\frown}\beta$. Below we only consider the latter case (the former follows analogously). We replace the derivation with one where the conclusion of the affected $\bot E$ has less size by the following transformation (denoted by $\rightsquigarrow$) of part of the derivation tree (the rest of the derivation tree is unchanged):

$$\dfrac{\begin{array}{c}[w:\alpha\breve{\frown}\beta\rightarrow\bot]^1\\ \Pi\\ v':\bot\end{array}}{w:\alpha\breve{\frown}\beta}\bot E^1 \quad\rightsquigarrow\quad \dfrac{\dfrac{\dfrac{[u:\beta\rightarrow\bot]^2 \qquad \dfrac{[w:\alpha\breve{\frown}\beta]^1[v:\alpha]^3[\mathcal{R}(v,u,w)]^3}{u:\beta}\breve{\frown}E}{u:\bot}\rightarrow E}{w:\bot}\bot E}{w:\alpha\breve{\frown}\beta\rightarrow\bot}\rightarrow I^1 \qquad \dfrac{\dfrac{\dfrac{\Pi}{v':\bot}\bot E^2}{u:\beta}\breve{\frown}I^3}{w:\alpha\breve{\frown}\beta} \ .$$

By induction it is now easy to see that repeated applications of this transformation yield the desired derivation. In the case of 2) we notice that if there is to be two $\perp E$ rules immediately following each other the uppermost has to have conclusion $v : \perp$ (for some $v$). But then it is clearly superfluous and can be removed, viz.

$$\begin{array}{c}[w : p \to \perp]^1 \\ \vdots \\ \dfrac{u : \perp}{v : \perp}\ \perp E \\ \dfrac{v : \perp}{w : p}\ \perp E^1\end{array} \quad \rightsquigarrow \quad \begin{array}{c}[w : p \to \perp]^1 \\ \vdots \\ \dfrac{u : \perp}{w : p}\ \perp E^1\end{array} .$$

Again, repeated applications give the desired derivation. □

In ordinary natural deduction the $\perp E$ rule is sometimes restricted to having a non-$\perp$ conclusion by definition [Pra65]. This would entail 2) above directly. But this restriction is *not* possible here as we must be able to propagate $\perp$ between different worlds to retain completeness.

The *major premise* of an E-rule is the premise containing the connective being eliminated. A premise which is not major is called *minor*.

**Definition 3.13** A *maximal formula* in a derivation is a labelled formula which is both the conclusion of an introduction rule and the major premise of an elimination rule. A derivation is *normal* if it contains no maximal formulas, all applications of $\perp E$ have atomic conclusions and there are no applications of $\perp E$ immediately following each other.

An introduced labelled formula which is immediately eliminated does clearly not contribute to the derivation, hence a maximal formula can be removed by a transformation called a *contraction step*. Below we show the case of $\smile$:

$$\begin{array}{c}[v' : \alpha]^1[\mathcal{R}(v', u', w)]^1 \\ \Pi \\ \dfrac{u' : \beta}{w : \alpha \smile \beta}\ \smile I^1 \qquad v : \alpha \quad \mathcal{R}(v, u, w) \\ \hline u : \beta\end{array} \smile E \quad \rightsquigarrow \quad \begin{array}{c} v : \alpha \quad \mathcal{R}(v, u, w) \\ \Pi[v/v', u/u'] \\ u : \beta\end{array} ,$$

where $\Pi[v/v', u/u']$ is obtained from $\Pi$ by systematically substituting $v$ for $v'$ and $u$ for $u'$, with a suitable renaming of the variables to avoid clashes.

**Theorem 3.14** *Any derivation can be transformed into a normal derivation.*

Proof. The restrictions on a normal derivation concerning $\perp E$ are taken care of by Proposition 3.12. Now, choose a largest (with respect to size) maximal formula $w : \alpha$ which has only maximal formulas of less size above it in the derivation and apply a contraction step. No new maximal formulas as large (or larger) than $w : \alpha$

are introduced by this step. Repeated applications of this step yield a derivation in normal form. □

**Definition 3.15** A *track* in a derivation $\Pi$ is a sequence of labelled formulas $w_1 : \alpha_1, w_2 : \alpha_2, \ldots, w_n : \alpha_n$ where $w_1 : \alpha_1$ is a leaf, and for $1 \leq i < n$, $w_{i+1} : \alpha_{i+1}$ is the conclusion of a rule of which $w_i : \alpha_i$ is a premise that is not a minor premise of $\rightarrow E$ or $\frown E$. A track of *order* 0 ends in the root of $\Pi$; a track of *order* $n+1$ ends in the minor premise of an E-rule with major premise belonging to a track of order $n$.

The above definition of a track is an extension of that of [Pra65] (we use the terminology of [TS96]) for propositional logic to $\mathcal{L}_{\widehat{\mathrm{AF}}}$. The key observation is that the structure of the rules $\rightarrow I$ and $\rightarrow E$ is similar to that of $\frown I$ and $\frown E$, respectively (disregarding judgments concerning the accessibility relation $\mathcal{R}$).

**Proposition 3.16** *Let $w_1 : \alpha_1, w_2 : \alpha_2, \ldots, w_n : \alpha_n$ be a track in a normal derivation. There is a* minimal formula *$\alpha_i$ such that*

1. *$w_j : \alpha_j$ (for all $j$, $1 \leq j < i$) is a major premise of an E-rule and $\alpha_{j+1}$ is a subformula of $\alpha_j$.*
2. *$w_i : \alpha_i$ ($i \neq n$) is a premise of an I-rule or $\bot E$.*
3. *$w_j : \alpha_j$ (for all $j$, $i < j < n$) is a premise of an I-rule and $\alpha_j$ is a subformula of $\alpha_{j+1}$.*

Proof. As the derivation is normal, in the track, an E-rule cannot follow an I-rule (there are no maximal formulas) and it cannot follow a $\bot E$ rule (the consequence is atomic). The $\bot E$ rule cannot follow an I-rule as the premise is $\bot$ and by normality there will thus at most be one $\bot E$ rule. □

If we let $\alpha \sqsubseteq \beta$ mean that $\alpha$ is a subformula of $\beta$ and $\alpha \sqsupseteq \beta$ that $\beta$ is a subformula of $\alpha$, the structure of a track can be illustrated as follows:

$$\alpha_1 \sqsupseteq \alpha_2 \sqsupseteq \ldots \sqsupseteq \alpha_j \sqsupseteq \ldots \sqsupseteq \alpha_i \sqsubseteq \ldots \sqsubseteq \alpha_{j'} \sqsubseteq \ldots \sqsubseteq \alpha_{n-1} \sqsubseteq \alpha_n.$$

**Definition 3.17** Consider a derivation $\Pi$ of $w : \alpha$ from $\Gamma$ and $\Delta$ in a system with no axioms. Let $S = \{\alpha\} \cup \{\gamma \mid u : \gamma \in \Gamma \text{ for some } u\}$. Then $\Pi$ is said to have the *subformula property* if for any labelled formula $v : \beta$ in $\Pi$, $\beta$ is

1. $\bot$,
2. a subformula of some formula in $S$, or
3. $\neg\beta'$ and $\beta'$ is a subformula of some formula in $S$.

**Theorem 3.18** *Any normal derivation satisfies the subformula property.*

PROOF. We start by observing that any formula in a derivation belongs to some track $w_1 : \alpha_1, w_2 : \alpha_2, \ldots, w_n : \alpha_n$. By Proposition 3.16, all $\alpha_i$ $(1 \leq i \leq n)$ are subformulas of either $\alpha_1$ or $\alpha_n$. By induction on the order of the track we now conclude that any formula in a normal derivation will be a subformula of either the root or a leaf. Consider a closed assumption: If it is closed by one of the I-rules it will be a subformula of the conclusion of that I-rule. If it is closed by $\bot E$ it will have the form $\neg\beta$ and $\beta$ will be the conclusion of that $\bot E$ rule. □

The results of this section could be proved for the $\{\bot, \rightarrow, \frown\}$ fragment as well but it would make the proofs more complicated. This corresponds to what Prawitz does when going from classical to intuitionistic logic in [Pra65] (it is not possible to restrict to a functionally complete set in intuitionistic logic).

Going even further, we could ask for a normalization result for the full set of connectives for $\mathcal{L}^{\frown}_{\mathrm{AF}}$, i.e., including $\wedge, \vee, \neg$. Such a normalization result is proved for FOL in [Stå91].

CHAPTER 4

# Interval Logic

This chapter introduces a number of interval logics. We will almost exclusively be concerned with interval logics that can be regarded as modal logics where the possible worlds are intervals. Furthermore, the interval logics we will concentrate on all include a notion for referring to the length of an interval.

We start by giving a thorough introduction to Signed Interval Logic (SIL) [Ras99a] in Section 4.1. We consider syntax, semantics and Hilbert proof system for SIL as well as a soundness and completeness result. In its most general form, the temporal domain of SIL has no ordering; we consider how the soundness and completeness result can be extended to cover a totally ordered domain. Furthermore, we sketch how SIL is related to arrow logic [MV97] and relational algebra [Tar41, SS93].

In Section 4.2 we consider a number of related interval logics. We most thoroughly discuss two interval logics very closely related to SIL, namely Interval Temporal Logic (ITL) [HMM83, Mos85, Dut95a] and Neighbourhood Logic (NL) [ZH98].

Finally, in Section 4.3 we consider Duration Calculus (DC) [ZHR91, HZ97] which extends interval logic with notions for accumulated durations of Boolean functions over intervals.

## 4.1 Signed Interval Logic

In this section we introduce Signed Interval Logic (SIL) [Ras99a]. Our presentation will build on the notions and concepts introduced in the previous chapters, in particular Chapter 3.

### 4.1.1 Syntax and Semantics

Syntactically, SIL is a first order $\mathcal{L}^{\frown}$ logic with equality. Additionally, we wish to distinguish function/predicate symbols as either *rigid* or *flexible*. In particular $=$ is rigid. Furthermore, SIL includes the rigid function symbols $0, -, +$ as well as the special flexible nullary function symbol $\ell$. We will denote the language containing these symbols together with the symbols of first order $\mathcal{L}^{\frown}$ a *signed interval language*. A formula/term is said to be *flexible* if it contains a flexible symbol. Otherwise it is said to be *rigid*. A formula is *chop-free* if it does not contain $\frown$.

We now turn to the semantics of SIL. As it is an $\mathcal{L}^{\frown}$ logic, it is given a semantics as discussed in Section 3.2. In particular, the semantics is given in terms of first order square models (with constant domains), cf. Definition 3.2 and Definition 3.4, with the important addition that the interpretation of a rigid symbol is the same in all worlds.[1]

In Definition 3.4, the set over which a square frame is defined is denoted $T$. We can now conveniently think of $T$ as a *temporal domain* and two temporal points of $T$ make up a *signed interval*, i.e., the possible worlds are signed intervals. For now we do not enforce any structure on $T$; we do not require it to be ordered in any way in particular. In the case of SIL we can have a completeness result without this requirement and therefore choose to define $T$ as general as possible. In Section 4.1.4 we will see how a total order can be enforced while still achieving completeness.

We want to be able to refer to the *signed length* of a signed interval. For this we define the following:

**Definition 4.1** Given a square frame $(W, R)$ over a set $T$, a *signed measure* is a function $m \in W \to D$ where $D$ is a set equipped with a binary operator $+$ and a distinguished element $0 \in D$. Furthermore, $m$ has to satisfy the following conditions for any $i, i', j, j' \in T$ and $a, b \in D$:

M1: if $m(i,j) = m(i,j')$ then $j = j'$,
if $m(j,i) = m(j',i)$ then $j = j'$,

M2: $m(i,i) = 0$,

M3: $m(i,j) + m(j,i') = m(i,i')$,

M4: $m(i,i') = a + b$ iff $m(i,k) = a$ and $m(k,i') = b$ for some $k \in T$.

We now define which properties a domain of values to represent signed lengths should have in general (some of these are implicitly given by the above definition).

**Definition 4.2** A *signed duration domain* is a group $(D, +, -, 0)$.[2]

[1] Hence, the interpretation of a flexible symbol can vary from world to world.
[2] The main binary operator of the group is $+$ and its unary inverse operator is $-$.

**Definition 4.3** A *signed interval model* is a square model where the domain is a signed duration domain and the interpretation of $\ell, +, -, 0$ is such that

$$I(\ell)(i,j) = m(i,j),$$

$$I(0)(i,j) = 0,$$

$$I(+)(i,j) = +,$$

$$I(-)(i,j) = -.$$

We say that a formula $\phi$ is *SIL-valid* (written $\models_{\mathrm{SIL}} \phi$) if it is valid in the class of all signed interval models.

### 4.1.2 Proof System

SIL is a first order associative normal logic with a binary modality (cf. Section 3.3) with the addition of the following axioms:

R: $\phi^\frown\psi \to \phi$ if $\phi$ is rigid,
$\phi^\frown\psi \to \psi$ if $\psi$ is rigid,

B: $((\exists x)\phi)^\frown\psi \to (\exists x)(\phi^\frown\psi)$ if $x \notin \mathrm{FV}(\psi)$,
$\phi^\frown((\exists x)\psi) \to (\exists x)(\phi^\frown\psi)$ if $x \notin \mathrm{FV}(\phi)$,

L1: $(\ell = s)^\frown\phi \to \neg((\ell = s)^\frown\neg\phi)$ if $s$ is rigid,
$\phi^\frown(\ell = s) \to \neg(\neg\phi^\frown(\ell = s))$ if $s$ is rigid,

L2: $\ell = s + t \leftrightarrow (\ell = s)^\frown(\ell = t)$ if $s$ and $t$ are rigid,

L3: $\phi \to \phi^\frown(\ell = 0)$,
$\phi \to (\ell = 0)^\frown\phi$.

Note how B corresponds to the Barcan formula mentioned in Section 3.3.

Furthermore, SIL contains axioms expressing the properties of a signed duration domain (cf. Definition 4.2):

D1: $(s + t) + u = s + (t + u)$,

D2: $s + 0 = s$,

D3: $s + (-s) = 0$.

There is one last (but very important) detail which has to be adressed: Just using the axiom Q1 of FOL "as is" would make SIL unsound. We have to slightly modify Q1 to accommodate the notions of rigidity and chop-freeness:

Q1: $(\forall x)\phi \rightarrow \phi[s/x]$ if $s$ is rigid or $\phi$ is chop-free.

We write $\vdash_{\mathrm{SIL}} \phi$ to denote theoremhood in this system for SIL.

Finally, we note that the system presented in this section is slightly different from the original system of [Ras99a]: First, the system is defined as an extension to a certain modal logic with a binary modality instead of being formulated directly. Second, the axioms L1,L2,D1,D2,D3 are formulated with arbitrary terms instead of (universally quantified) variables. These small changes make later developments simpler, both theoretically and practically. It is trivial to show that the formulations are equivalent.

### 4.1.3 Soundness and Completeness

In this section we sketch the proof of a completeness result for SIL: The proof system of SIL is sound and complete with respect to the class of all signed interval models. We refer to [Ras99b] for a full, detailed proof.

The proof of completeness follows the general structure of a Henkin-style completeness proof [Men87, HC68]. The central idea in a Henkin-style proof is the following: Given an arbitrary formula $\phi$ which is not a theorem, construct a model which satisfies $\neg\phi$. This implies the non-validity of $\phi$ and the completeness follows.

We start by presenting some standard concepts and results used in Henkin-style completeness proofs, namely results concerning maximal consistent sets and witnesses [Gar84, HC68, Ham88].

**Definition 4.4** Let $\Gamma$ be a set of closed formulas of a signed interval language $L$.

- $\Gamma$ is *consistent* (with respect to SIL) if there is no finite subset $\{\phi_1, \ldots, \phi_n\}$ of $\Gamma$ such that $\vdash_{\mathrm{SIL}} \neg(\phi_1 \wedge \ldots \wedge \phi_n)$.
- $\Gamma$ is *maximally consistent* if it is consistent and there is no consistent set of closed formulas $\Gamma'$ such that $\Gamma \subset \Gamma'$.

Let $B = \{b_0, b_1, b_2, \ldots\}$ be an infinite, countable set of symbols not occurring in the signed interval language $L$. Let $L^+$ denote the signed interval language obtained by adding all symbols of $B$ to $L$ as rigid constants.

**Definition 4.5** A set $\Gamma$ of closed formulas of $L^+$ is said to have *witnesses in* $B$ if for every closed formula of $\Gamma$ of the form $(\exists x)\phi$ (where $x$ is the only free variable of $\phi$) there exists a constant $b_i \in B$ such that $\phi[b_i/x] \in \Gamma$.

**Theorem 4.6** *If $\Gamma$ is a consistent set of closed formulas of $L$, there is a set $\Gamma^*$ of closed formulas of $L^+$ which satisfies the following:*

- $\Gamma \subseteq \Gamma^*$,
- $\Gamma^*$ *is maximally consistent,*

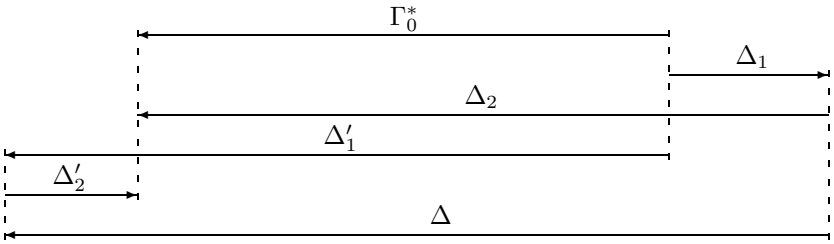



Figure 4.1: Possible configuration of the worlds of Proposition 4.8

- $\Gamma^*$ *has witnesses in* $B$.

If $\Gamma_0$ is a consistent set of closed formulas of $L$, let $\Gamma_0^*$ be a set of closed formulas of $L^+$ whose existence is guaranteed by the above theorem. We denote by $\Sigma_0$ the set of rigid closed formulas of $\Gamma_0^*$.

Given a consistent set $\Gamma_0$ of closed formulas we can now construct a model $\mathfrak{M}_0 = (W_0, R_0, D_0, I_0)$ where the worlds of $W_0$ are certain maximal consistent sets of closed formulas (including $\Gamma_0^*$), $R_0$ is defined by:

$$R_0(\Delta_1, \Delta_2, \Delta) \quad \text{iff} \quad \text{if } \phi_1 \in \Delta_1 \text{ and } \phi_1 \in \Delta_2 \text{ then } (\phi_1{}^\frown\phi_2) \in \Delta \text{ for any } \phi_1, \phi_2,$$

and $D_0$ is the set of equivalence classes with respect to $=$ over $B$. Finally, $I_0$ is defined for all symbols in all worlds. For example, in the case of a propositional letter we define $I_0(p)(\Delta) = (p \in \Delta)$, i.e. $\mathfrak{M}_0, \mathcal{V}, \Delta \models p$ iff $p \in \Delta$.

The following theorem generalizes the case of a propositional letter to arbitrary formulas [Dut95a].

**Theorem 4.7**

$$\mathfrak{M}_0, \mathcal{V}, \Delta \models \phi \quad \textit{iff} \quad \phi \in \Delta.$$

The model $\mathfrak{M}_0$ will play a central part in the construction of a satisfying signed interval model. Another important part in this construction will be played by the following proposition.

**Proposition 4.8** *Let* $((\Delta_1, \Delta_2), (\Delta_1', \Delta_2')) \in (W_0 \times W_0) \times (W_0 \times W_0)$. *Then, if* $R_0(\Delta_1, \Delta_2, \Gamma_0^*)$ *and* $R_0(\Delta_1', \Delta_2', \Gamma_0^*)$, *there is a unique world* $\Delta \in W_0$ *such that* $R_0(\Delta_1, \Delta, \Delta_1')$ *and* $R_0(\Delta, \Delta_2', \Delta_2)$.

The intuition of this proposition can be given in terms of signed intervals: Given a pair of pairs $((\Delta_1, \Delta_2), (\Delta_1', \Delta_2'))$ of consecutive signed intervals of the current signed interval $\Gamma_0^*$ there is a unique signed interval $\Delta$ lying between the two chopping points of the two pairs of signed intervals. We have sketched this intuition in Figure 4.1.

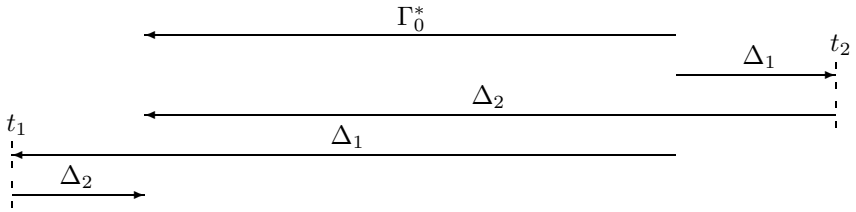



Figure 4.2: Intuitively, the points of $T$ are the "chopping points" (marked by $t_1$ and $t_2$ on the figure) of the pairs $(\Delta_1, \Delta_2)$ related by $R_0(\Delta_1, \Delta_2, \Gamma_0^*)$. The figure shows two of the possible pairs $(\Delta_1, \Delta_2)$.

We will now begin constructing a signed interval model from $\mathfrak{M}_0$. For this we need to define a temporal domain $T$:

$$T = \{\ (\Delta_1, \Delta_2) \in W_0 \times W_0 \mid R_0(\Delta_1, \Delta_2, \Gamma_0^*)\ \}.$$

The intuition behind this particular definition of $T$ is the following: If we think of the worlds of $W_0$ as signed intervals, $T$ is the set of all pairs of consecutive signed intervals of the current signed interval. These pairs define, by means of their chopping points, all the necessary temporal points. See Figure 4.2.

We have now come to the crucial step in the construction. Intuitively, we want to identify a signed interval given by two points of $T$ with a signed interval of $W_0$. But this connection is exactly what Proposition 4.8 gives us. Formally, let $\mu : T \times T \to W_0$ such that $\mu((\Delta_1, \Delta_2), (\Delta_1', \Delta_2'))$ is the world $\Delta$ given by Proposition 4.8. Revisit Figure 4.1 for the intuition.

We are now ready to construct a model $\mathfrak{M} = (W, R, D, I)$ on the basis of $\mathfrak{M}_0$ as follows:

- The frame $(W, R)$ is the square frame defined by $T$.
- The domain $D$ is the same as $D_0$.
- The interpretation function $I$ is defined by $I(s)(i,j) = I_0(s)(\mu(i,j))$ for any symbol $s$ and any signed interval $(i,j)$.

The following proposition shows that $\mathfrak{M}$ indeed is a signed interval model.

**Proposition 4.9** *The constructed model $\mathfrak{M}$ is a signed interval model.*

Proof. We have to make sure that $D$ of $\mathfrak{M}$ is a signed duration domain and that the interpretation is as specified in Definition 4.3.

The rigid symbols $-$ and $+$ of $L$ define a unary and a binary operation we also denote by $-$ and $+$ in $D$. The interpretation of the rigid constant $0$ will be an element of $D$ we also denote by $0$. As D1–D3 are valid in $\mathfrak{M}_0$ they will by Theorem 4.12 (see below) also be valid in $\mathfrak{M}$, hence $(D, +, -, 0)$ is a signed duration domain. We now

only need an appropriate signed measure. But it can be shown [Ras99b] that the interpretation of $\ell$ is already defined such that M1–M4 of Definition 4.1 are satisfied. Thus, we define the signed measure by $m(i,j) = I(\ell)(i,j)$ for any signed interval $(i,j) \in W$. □

We want to establish a connection between satisfaction of formulas in $\mathfrak{M}$ and $\mathfrak{M}_0$: We want to show that a formula is satisfied in a world $(i,j)$ of $\mathfrak{M}$ iff it is satisfied in the corresponding world $\mu(i,j)$ of $\mathfrak{M}_0$. The only difficulty is in the case of chop. For this we need the following two propositions (see [Ras99b] for proofs).

**Proposition 4.10** *If $i,j,k \in T$ then $R_0(\mu(i,k), \mu(k,j), \mu(i,j))$.*

**Proposition 4.11** *Let $i,j \in T$ and $\Gamma_1, \Gamma_2 \in W_0$. If $R_0(\Gamma_1, \Gamma_2, \mu(i,j))$ then $\Gamma_1 = \mu(i,k)$ and $\Gamma_2 = \mu(k,j)$ for some $k \in T$.*

We can now formulate the connection between $\mathfrak{M}$ and $\mathfrak{M}_0$. We note that since the domains of $\mathfrak{M}$ and $\mathfrak{M}_0$ are the same, an $\mathfrak{M}$-valuation is also an $\mathfrak{M}_0$-valuation.

**Theorem 4.12**

$$\mathfrak{M}, \mathcal{V}, (i,j) \models \phi \quad \textit{iff} \quad \mathfrak{M}_0, \mathcal{V}, \mu(i,j) \models \phi.$$

Proof. The proof is by a straightforward structural induction over $\phi$: In the case of $\phi$ being $\psi^\frown\chi$ we use Propositions 4.10 and 4.11. See [Ras99b]. □

We can now establish the main result of this section.

**Theorem 4.13** *If $\Gamma_0$ is a consistent set of closed formulas (with respect to SIL) then we can construct a signed interval model which satisfies $\Gamma_0$.*

Proof. We know by Proposition 4.9 that the constructed model $\mathfrak{M}$ is a signed interval model. We are therefore done if we can show that $\mathfrak{M}$ satisfies $\Gamma_0$.

It is possible to find two worlds $\Delta_1, \Delta_2 \in W_0$ such that $R_0(\Delta_1, \Gamma_0^*, \Gamma_0^*)$ and $R_0(\Gamma_0^*, \Delta_2, \Gamma_0^*)$ (see [Ras99b]). Thus, both $i = (\Delta_1, \Gamma_0^*)$ and $j = (\Gamma_0^*, \Delta_2)$ belong to $T$, hence $(i,j) \in W$. It is now immediate (by definition of $\mu$) that $\mu(i,j) = \Gamma_0^*$. Then, utilizing Theorems 4.12 and 4.7, we are done. □

From the above theorem the completeness of SIL follows easily.

**Theorem 4.14** *A formula $\phi$ is valid in the class of all signed interval models iff it is a theorem of SIL,*

$$\models_{\text{SIL}} \phi \quad \textit{iff} \quad \vdash_{\text{SIL}} \phi.$$

PROOF. For the *if*-part (soundness) we simply have to check that all axioms of SIL are valid in the class of all signed interval models and that all inference rules of SIL preserve validity. This is straightforward.

For the *only if*-part (completeness) assume $\phi$ is *not* a theorem of SIL. We now have to show that $\phi$ is not valid in some signed interval model. Let $\phi'$ be the universal closure of $\phi$; $\phi'$ is not a theorem either. The set $\{\neg\phi'\}$ will therefore be consistent and we can construct a signed interval model $\mathfrak{M}$ which satisfies $\neg\phi'$ (Theorem 4.13). Since $\neg\phi'$ is satisfied by $\mathfrak{M}$, $\phi'$ is not valid in $\mathfrak{M}$ and neither is $\phi$. □

**REMARK 4.15** The above soundness and completeness result is (in some sense) the most general possible with the signed duration domain just being a group. It will later turn out convenient if we require the signed duration domain to be commutative, i.e., an Abelian group. It is trivial to modify the completeness result in this case. Simply add an axiom to the Hilbert system stating commutiativity,

D4: $s + t = t + s$,

and the rest follows straightforwardly. In fact, henceforth, when referring to SIL we will implicitly assume this Abelian version.

### 4.1.4 Totally Ordered SIL

The completeness result of the previous section is for a general class of signed interval models with no ordering on the underlying domains $T$ and $D$. To justify the name "interval logic" one could argue that it would be more natural to require a total ordering on these domains. We will in this section show how a completeness result can be established in this case, based on [Ras99c]. Note that this is not as simple as in the Abelian case discussed above, as we here have to relate the two orderings on the domains $T$ and $D$.

It turns out that such a completeness result for *Signed Interval Logic on Totally Ordered Domains* ($\text{SIL}_{\text{to}}$) can be established fairly easily. Furthermore, this can be done in a conservative way, in the sense that the class of signed interval models just can be restricted by requiring total orderings on the underlying domains. The proof system can correspondingly be modified simply by adding further axioms.

**DEFINITION 4.16** A *totally ordered temporal domain* $(T, \leq)$ is a non-empty set $T$ together with a total order $\leq$ on $T$.

**DEFINITION 4.17** A *totally ordered signed measure* is an extension of a signed measure (Defintion 4.1) by requiring $D$ to be equipped with a binary predicate $\leq$ and $m$ to satisfy the following condition as well (besides M1–M4):

M5: $m(i, j) \leq 0$ iff $j \leq i$.

This condition relates the total ordering of the temporal domain to the total ordering of the duration domain.

**Definition 4.18** A *totally ordered signed duration domain* is a totally ordered group $(D, +, -, 0, \leq)$. Thus, $(D, +, -, 0)$ is a group, $\leq$ is a total order on $D$, and the following monotonicity rule is satisfied for all $a, b, c \in D$:

$$a \leq b \rightarrow (a + c \leq b + c) \wedge (c + a \leq c + b).$$

This notion of a totally ordered group follows the classical way of extending a group with a total order [Fuc63].

The syntax of $\mathrm{SIL}_{\mathrm{to}}$ is that of SIL extended with the rigid binary predicate $\leq$.

**Definition 4.19** A *totally ordered signed interval model* is a signed interval model as in Definition 4.3 where the temporal domain, the signed duration domain and the signed measure are of the totally ordered versions defined above and the binary predicate $\leq$ is interpreted the obvious way.

We say that a formula $\phi$ is $\mathrm{SIL}_{\mathrm{to}}$-*valid* (written $\models_{\mathrm{SIL}_{\mathrm{to}}} \phi$) if it is valid in the class of all totally ordered signed interval models

As mentioned above, the proof system of $\mathrm{SIL}_{\mathrm{to}}$ is a conservative extension (in the sense of adding further axioms) to the proof system of SIL.

These further axioms are D5–D9 (listed below), which together with D1–D3 express the properties of Definition 4.18.

D5: $s \leq s$,

D6: $s \leq t \rightarrow (t \leq u \rightarrow s \leq u)$,

D7: $s \leq t \wedge t \leq s \rightarrow s = t$,

D8: $s \leq t \vee t \leq s$,

D9: $s \leq t \rightarrow s + u \leq t + u$,
$s \leq t \rightarrow u + s \leq u + t$.

**Soundness and Completeness of** $\mathrm{SIL}_{\mathrm{to}}$

In this section we give a soundness and completeness proof for $\mathrm{SIL}_{\mathrm{to}}$. The proof extends the proof for SIL and we here only consider necessary extensions and modifications to the existing proof.

The first extension necessary is in connection with the model construction where we need to define a temporal domain $T$. We define $T$ as in Section 4.1.3:

$$T = \{\ (\Delta_1, \Delta_2) \in W_0 \times W_0 \mid R_0(\Delta_1, \Delta_2, \Gamma_0^*)\ \},$$

but we now also need a total order $\leq$ on $T$. This can be defined the following way (where $(\Delta_1, \Delta_2), (\Delta_1', \Delta_2') \in T$):

$$(\Delta_1, \Delta_2) \leq (\Delta_1', \Delta_2') \quad \text{iff} \quad \begin{cases} (\ell = b) \in \Delta_1, \\ (\ell = b') \in \Delta_1', \\ (b \leq b') \in \Sigma_0, \end{cases}$$

for some $b, b' \in B$. It is straightforward to see that this definition will give a total order on $T$: As D5–D8 are axioms of $\mathrm{SIL}_{\mathrm{to}}$ they will by construction be members of $\Sigma_0$ and the order relation $\leq$ on the righthand side will therefore be a total order. This implies that the order relation on $T$ will be total too.

As in Section 4.1.3 we construct a model $\mathfrak{M}$ on the basis of the model $\mathfrak{M}_0$.

**Proposition 4.20** *The constructed model $\mathfrak{M}$ is a totally ordered signed interval model.*

Proof. The proof follows the exact same lines as that of Proposition 4.9. The only new case is that we have to show that M5 is satisfied when we define $m(i,j) = I(\ell)(i,j)$.

For this, let $i = (\Delta_1, \Delta_2)$ and $j = (\Delta_1', \Delta_2')$ be arbitrary members of $T$. M5 states the condition:

$$m(i,j) \leq 0 \quad \text{iff} \quad j \leq i. \tag{4.1}$$

We start by expanding the left- and righthand sides by definitions. First the lefthand side:

$$m(i,j) = I(\ell)(i,j) = I_0(\ell)(\mu(i,j)) = I_0(\ell)(\mu((\Delta_1, \Delta_2), (\Delta_1', \Delta_2'))).$$

By definition of $\mu$ there is a unique $\Delta \in W_0$ such that

$$I_0(\ell)(\mu((\Delta_1, \Delta_2), (\Delta_1', \Delta_2'))) = I_0(\ell)(\Delta),$$

and

$$R_0(\Delta_1, \Delta, \Delta_1'). \tag{4.2}$$

As $(\exists x)(\ell = x)$ is a theorem it will by construction belong to the maximal consistent set $\Delta$. But then there is a witness $a \in B$ such that $(\ell = a) \in \Delta$. By definition this means that $I_0(\ell)(\Delta) = [a]$, where $[a]$ is the equivalence class with representative $a$. (Remember how the domain $D_0$ was defined in Section 4.1.3.) Hence, the lefthand side of (4.1) amounts to $[a] \leq 0$. As $a$, $\leq$ and $0$ are rigid this is by definition equivalent to:

$$(a \leq 0) \in \Sigma_0. \tag{4.3}$$

We now turn to the righthand side of (4.1):

$$j \leq i \quad \text{iff} \quad (\Delta_1', \Delta_2') \leq (\Delta_1, \Delta_2) \quad \text{iff} \quad \begin{cases} (\ell = b) \in \Delta_1, \\ (\ell = b') \in \Delta_1', \\ (b' \leq b) \in \Sigma_0, \end{cases}$$

for some $b, b' \in B$. The conditions $(\ell = b) \in \Delta_1$ and $(\ell = b') \in \Delta_1'$ are by construction satisfied for some $b, b' \in B$, hence the righthand side of (4.1) amounts to $(b' \leq b) \in \Sigma_0$. In other words, to show (4.1) we have to show (cf. (4.3)):

$$(a \leq 0) \in \Sigma_0 \quad \text{iff} \quad (b' \leq b) \in \Sigma_0. \tag{4.4}$$

For this we need a connection between $a$ and $b, b'$. But this is exactly what (4.2) gives us. As $(\ell = b) \in \Delta_1$ and $(\ell = a) \in \Delta$ we have by definition of $R_0$ that $(\ell = b)^\frown(\ell = a) \in \Delta_1'$. By axiom L2 we then deduce that $(\ell = b + a) \in \Delta_1'$. As $(\ell = b') \in \Delta_1'$ we conclude that $(b + a = b') \in \Delta_1'$. Because of rigidity we have $(b + a = b') \in \Sigma_0$ and utilizing the fact that D1–D9 are axioms of $\mathrm{SIL_{to}}$ it is now straightforward to show (4.4) and thus (4.1). In particular, notice that the monotonicity axiom D9 is needed here. □

Propositions 4.10 and 4.11, and Theorems 4.12 and 4.13 do not assume any properties of $T$ and are therefore immediately usable here.

We can thus formulate the main result of this section (the proof is similar to that of Theorem 4.14).

**Theorem 4.21** *A formula $\phi$ is valid in the class of all totally ordered signed interval models iff it is a theorem of* $\mathrm{SIL_{to}}$,

$$\models_{\mathrm{SIL_{to}}} \phi \quad \textit{iff} \quad \vdash_{\mathrm{SIL_{to}}} \phi.$$

In $\mathrm{SIL_{to}}$, the following abbreviations will often be used:

$$\begin{aligned} \mathit{fwd} &\mathrel{\widehat{=}} 0 \leq \ell, \\ \mathit{bwd} &\mathrel{\widehat{=}} \ell \leq 0. \end{aligned}$$

### 4.1.5 Arrow Logic and Relational Algebra

In this section we consider the expressiveness of SIL by relating it to *arrow logic* [MV97] and *relational algebra* [Tar41, SS93].[3]

The results rely on the capability to define an abbreviated unary modality $^{-1}$ (read: converse) in SIL which "reverses" the direction of an interval:

$$\phi^{-1} \quad \widehat{=} \quad (\exists x)(\ (\ell = x) \wedge (\ (\ell = 0) \wedge (\ell = x)^\frown \phi\ )^\frown \mathrm{true}\ ),$$

[3] It should be noted that none of the results of this section require totally ordered domains as considered in the previous section.

where $x$ is some variable not free in $\phi$. One can straightforwardly semantically show that $\phi^{-1}$ is satisfied on a signed interval $(i,j)$ iff $\phi$ is satisfied on the signed interval $(j,i)$.

Arrow logic [MV97] is a propositional modal logic where the possible worlds are pairs of elements from some set. We see that this corresponds to signed intervals of SIL which makes a comparison interesting.

Arrow logic is equipped with a constant $\iota\delta$, a unary modality $\otimes$ and a binary modality $\circ$. The semantics of $\iota\delta$, $\otimes$ and $\circ$ can informally be given in terms of SIL: $\iota\delta$ corresponds to $(\ell = 0)$, $\otimes$ to $^{-1}$, and $\circ$ to $^\frown$. In arrow logic $\iota\delta$, $\otimes$ and $\circ$ are basic modalities and not abbreviations of some kind. In SIL we can define $^{-1}$ using $^\frown$ and $\ell$. Thus, we conclude that SIL can express the same as arrow logic with just the basic modality $^\frown$ (corresponding to $\circ$) and then the special symbol $\ell$.

But this expressive power of SIL has a price: Firstly, the introduction of $\ell$ restricts the set of possible models. Secondly, to define $^{-1}$ it was necessary to use a first order construct to quantify over the value of $\ell$. In conclusion we can therefore (rather informally) state that: SIL is a first order arrow logic with the special unary function symbol $\ell$.

We now consider another consequence of $^{-1}$: It seems natural to think of signed intervals as *binary relations*, hence $(i,j) \in T \times T$ asserts that $i$ is related to $j$. In some signed interval model, a formula $\phi$ will either be true or false on some signed interval. If we now consider the set of all signed intervals on which $\phi$ is true we will have a binary relation on $T$. We will in the following pursue this idea by relating SIL to relational algebra [SS93].

**Definition 4.22** A *relational algebra* $\mathfrak{RA} = \langle \mathcal{S}, \oplus, \odot, \circ, \ominus, \otimes, \mathbf{0}, \mathbf{1}, \iota\delta \rangle$ is a non-empty set $\mathcal{S}$ equipped with three binary operators $\oplus, \odot, \circ$, two unary operators $\ominus, \otimes$, and three constants $\mathbf{0}, \mathbf{1}, \iota\delta$ such that $\langle \mathcal{S}, \oplus, \odot, \ominus, \mathbf{0}, \mathbf{1} \rangle$ is a Boolean algebra and the following axioms are satisfied for all $x, y, z \in S$:

| | | | |
|---|---|---|---|
| RA1 | $(x \oplus y) \circ z \;=\; (x \circ z) \oplus (y \circ z)$, | RA5 | $x \circ \iota\delta \;=\; x$, |
| RA2 | $\otimes(x \oplus y) \;=\; \otimes x \oplus \otimes y$, | RA6 | $\otimes\otimes x \;=\; x$, |
| RA3 | $(x \circ y) \circ z \;=\; x \circ (y \circ z)$, | RA7 | $\otimes(x \circ y) \;=\; \otimes y \circ \otimes x$. |
| RA4 | $\ominus(\otimes x \circ \ominus(x \circ y)) \oplus \ominus y \;=\; \mathbf{1}$, | | |

We now formally define how to build the above mentioned binary relations.

**Definition 4.23** Let $\mathfrak{M} = (W, R, D, I)$ be a signed interval model and $\mathcal{V}$ be a $\mathfrak{M}$-valuation. As $\mathfrak{M}$ is a signed interval model we have $W = T \times T$ for some set $T$. We now define a *SIL-relation* over $\phi$ (written $\mathrm{R}_{\mathfrak{M},\mathcal{V}}(\phi)$) by:

$$\mathrm{R}_{\mathfrak{M},\mathcal{V}}(\phi) = \{(i,j) \in T \times T \mid \mathfrak{M}, \mathcal{V}, (i,j) \models \phi\}.$$

Furthermore, we define the set $\mathcal{R}_{\mathfrak{M},\mathcal{V}}$ of all SIL-relations in a given model and valuation:

$$\mathcal{R}_{\mathfrak{M},\mathcal{V}} = \{\mathrm{R}_{\mathfrak{M},\mathcal{V}}(\phi) \mid \phi \text{ is a SIL formula}\}.$$

To establish the connection to relational algebra, we associate three binary operators $+, \cdot, ;$, two unary operators $-, \sim$ and three constants $0, 1, 1'$ with $\mathcal{R}_{\mathfrak{M},\mathcal{V}}$. The meaning of these operators and constants is given by the following equivalences:

$$\begin{array}{rclrcl} 1 & = & \mathrm{R}_{\mathfrak{M},\mathcal{V}}(\mathrm{true}), & -\mathrm{R}_{\mathfrak{M},\mathcal{V}}(\phi) & = & \mathrm{R}_{\mathfrak{M},\mathcal{V}}(\neg\phi), \\ 0 & = & \mathrm{R}_{\mathfrak{M},\mathcal{V}}(\mathrm{false}), & \mathrm{R}_{\mathfrak{M},\mathcal{V}}(\phi) + \mathrm{R}_{\mathfrak{M},\mathcal{V}}(\psi) & = & \mathrm{R}_{\mathfrak{M},\mathcal{V}}(\phi \vee \psi), \\ 1' & = & \mathrm{R}_{\mathfrak{M},\mathcal{V}}(\ell = 0), & \mathrm{R}_{\mathfrak{M},\mathcal{V}}(\phi) \cdot \mathrm{R}_{\mathfrak{M},\mathcal{V}}(\psi) & = & \mathrm{R}_{\mathfrak{M},\mathcal{V}}(\phi \wedge \psi), \\ \sim\mathrm{R}_{\mathfrak{M},\mathcal{V}}(\phi) & = & \mathrm{R}_{\mathfrak{M},\mathcal{V}}(\phi^{-1}), & \mathrm{R}_{\mathfrak{M},\mathcal{V}}(\phi); \mathrm{R}_{\mathfrak{M},\mathcal{V}}(\psi) & = & \mathrm{R}_{\mathfrak{M},\mathcal{V}}(\phi^\frown\psi). \end{array}$$

Any SIL-relation built using any of the three constants and five operators can thus by simple equational reasoning be transformed to a single SIL-relation $\mathrm{R}_{\mathfrak{M},\mathcal{V}}(\phi)$ for some formula $\phi$.

To show equivalence of SIL-relations we have the following lemma.

**Lemma 4.24** $\models_{\mathrm{SIL}} \phi \leftrightarrow \psi$ *implies* $\mathrm{R}_{\mathfrak{M},\mathcal{V}}(\phi) = \mathrm{R}_{\mathfrak{M},\mathcal{V}}(\psi)$.

We can now formulate the following theorem saying that $\mathcal{R}_{\mathfrak{M},\mathcal{V}}$ together with the above defined operators and constants is a relational algebra.

**Theorem 4.25** $\mathfrak{R}_{\mathrm{SIL}} = \langle \mathcal{R}_{\mathfrak{M},\mathcal{V}}, +, \cdot, ;, -, \sim, 0, 1, 1' \rangle$ *is a relational algebra.*

Proof. By Definition 4.22 we must first show that $\langle \mathcal{R}_{\mathfrak{M},\mathcal{V}}, +, \cdot, -, 0, 1 \rangle$ is a Boolean algebra. But this follows easily due to the standard correspondence between propositional logic and Boolean algebra.

We must then show that $\mathfrak{R}_{\mathrm{SIL}}$ satisfies the axioms RA1–RA7 for arbitrary members of $\mathcal{R}_{\mathfrak{M},\mathcal{V}}$. For this we use Lemma 4.24. For example, we can show that $\mathfrak{R}_{\mathrm{SIL}}$ satisfies RA6 by showing $\models_{\mathrm{SIL}} (\phi^\frown\psi)^{-1} \leftrightarrow \psi^{-1}{}^\frown\phi^{-1}$. But this is not difficult; see [Ras99b] for a full proof. □

Theorem 4.25 gives a nice theoretical characterization of the expressive power of SIL. It is only establishable because $^{-1}$ is definable in SIL.

The results of this section are further elaborated in [Ras99b].

## 4.2 Related Interval Logics

In this section we consider a selection of interval logics of which some are closely related to SIL whereas others have more loose ties. Importantly, none of the interval logics of this section incorporate the notion of a direction of an interval.

We start by considering Interval Temporal Logic (ITL) which was probably the first interval logic to be introduced in computer science. ITL was originally based on discrete temporal domains and we discuss some initial work in this direction. Then we consider a more contemporary version based on general temporal domains; here the connection to SIL becomes very clear. Following this, we consider Neighbourhood Logic (NL) which is also closely connected to both (the general) ITL and SIL. Finally, in the last part, we briefly survey some other interval logics.

### 4.2.1 Interval Temporal Logic

#### Discrete ITL

We will here discuss various work on ITL with the common denominator that intervals are taken to be finite sequences $s_0, s_1, \ldots, s_n$ of states, thus time is considered discrete.

ITL in this setting was introduced in [HMM83, Mos85] and was used to specify timing aspects of hardware components and to reason about such specifications.

Both [HMM83] and [Mos85] introduce a chop modality ; with the following semantics (where $\phi_1$ and $\phi_2$ are arbitrary formulas of the respective logics):

$$s_0, s_1, \ldots, s_n \models \phi_1; \phi_2 \quad \text{iff} \quad s_0, s_1, \ldots, s_i \models \phi_1 \text{ and } s_i, s_{i+1}, \ldots, s_n \models \phi_2,$$

for some $i$ where $0 \leq i \leq n$. Furthermore, [HMM83] introduces the special symbol *len* which for any interval $s_0, s_1, \ldots, s_n$ equals the length $n$.

In [HMM83, Mos85] ITL is considered in a purely semantic setting. In [RP86] there is a Hilbert system for a version of ITL called *choppy logic*, having a chop modality with semantics as ; above.[4] The proof system includes, e.g., an axiom stating the associativity of ;. There is a completeness result for choppy logic in [RP86] and decidability is also considered.

In [Mos94] ITL is extended in different ways, e.g., by introducing sorts such that every variable of the logic is associated with a sort. A proof system for this extended ITL (based on the proof system for choppy logic [RP86]) is presented in [Mos94] and is shown to be sound and complete. The logic is used for specification and reasoning of various examples of concurrent systems.

In [Mos95] the work in [Mos94] is further extended by introducing the notion of temporal projection which means that only a subset of the states (the projected states) of an interval is considered when evaluating a formula. Also the possibility of intervals as infinite sequences of states is incorporated into the logic. The proof system is extended accordingly to accommodate these extensions.

#### General ITL

We will now consider a more general version of ITL based on abstractly defined temporal domains. This version of ITL was introduced in [Dut95a].

ITL is in this setting very similar to SIL. The syntax is the same whereas the semantics is slightly modified: Only intervals $(i, j)$ where $i \leq j$ are considered. Hence, the semantics is based on a restricted class of square models with a total order on the temporal domain. To emphasize the fact that $i \leq j$ is required we will for convenience denote these intervals by $[i, j]$. The semantics of $\frown$ in ITL can now be given as:

$$\mathfrak{M}, [i, j] \models \phi \frown \psi \quad \text{iff} \quad \mathfrak{M}, [i, k] \models \phi \text{ and } \mathfrak{M}, [k, j] \models \psi \text{ for some } k \text{ with } i \leq k \leq j.$$

[4] Actually, choppy logic is defined such that intervals can be both finite and infinite sequences of states.

This is the most important difference to SIL. Some modifications to the definition of measure and duration domain are necessary too. For a further discussion of similarities and difference of the semantics of ITL and SIL we refer to [Ras99b].

The proof systems are quite similar too: Only the axioms relating to the duration domain distinguish the two systems: In ITL the axioms D1–D3 are not present; instead five other related axioms are added. We again refer to [Ras99b] for further elaboration.

There is an alternative way of defining the proof system for ITL on the basis of that for $\mathrm{SIL}_{\mathrm{to}}$ (cf. [Dut95b]): Simply add the axiom $0 \leq \ell$ and modify the axiom L2 of $\mathrm{SIL}_{\mathrm{to}}$ such that it reads:

$$0 \leq s \wedge 0 \leq t \rightarrow (\ell = s + t \leftrightarrow (\ell = s)^\frown(\ell = t)),$$

where $s$ and $t$ are rigid.

In [Dut95a, Dut95b] a soundness and completeness result for ITL is presented.

We now consider the question of "imitating" ITL in SIL. Firstly, have to assume the totally ordered version of SIL, $\mathrm{SIL}_{\mathrm{to}}$; otherwise we cannot express the semantics of ITL. Secondly, we have to decide what "to do" with the direction of an interval. One possibility is to regard the pairs $(i, j)$ and $(j, i)$ as representing the same interval. This would technically be a little difficult to handle, so we choose instead to solely restrict attention to forward intervals (i.e., intervals where $0 \leq \ell$) of $\mathrm{SIL}_{\mathrm{to}}$. By this we mean that we assume the current interval to be forward and we make sure that we can only reach forward intervals. Given this, the contracting chop modality of ITL can be defined in $\mathrm{SIL}_{\mathrm{to}}$ as follows:

$$\phi \,|^\frown|\, \psi \quad \widehat{=} \quad (\phi \wedge \mathit{fwd})^\frown(\psi \wedge \mathit{fwd}).$$

We will not go further into this here but refer instead to Section 6.3 where we will see how the principle works in a system more suitable for conducting proofs of ITL in $\mathrm{SIL}_{\mathrm{to}}$.

### 4.2.2 Neighbourhood Logic

As mentioned in Section 1.5, ITL can not express unbounded liveness properties.

A logic which is able to specify such liveness properties is Neighbourhood Logic (NL) [ZH98]. NL can be seen as extending the general ITL. Syntactically, instead of the binary $^\frown$, NL incorporates two unary modalities $\Diamond_l$ (left neighbourhood) and $\Diamond_r$ (right neighbourhood). Semantically, the principles are the same as for ITL, in that formulas are interpreted on intervals $[i, j]$ (where $i \leq j$). We here give the semantics of the two unary modalities and refer to [ZH98] for a detailed discussion:

$$\begin{array}{lll} \mathfrak{M}, [i,j] \models \Diamond_r \phi & \text{iff} & \mathfrak{M}, [j,k] \models \phi \text{ for some } k \text{ with } j \leq k, \\ \mathfrak{M}, [i,j] \models \Diamond_l \phi & \text{iff} & \mathfrak{M}, [k,i] \models \phi \text{ for some } k \text{ with } k \leq i. \end{array}$$

A complete Hilbert system for NL is presented in [ZH98, BZ97]. The proof of completeness is given in [BZ97] and is based on the proof for ITL in [Dut95a].

As for ITL, we show how NL can be "imitated" in $\mathrm{SIL}_{\mathrm{to}}$ by defining the two modalities ($\Diamond_l$ and $\Diamond_r$) of NL as abbreviated modalities in $\mathrm{SIL}_{\mathrm{to}}$:

$$\begin{array}{lcl} \overline{\Diamond}_l\phi & \widehat{=} & ((\ell = 0) \wedge bwd^\frown\phi)^\frown\mathrm{true}, \\ \overline{\Diamond}_r\phi & \widehat{=} & \mathrm{true}^\frown((\ell = 0) \wedge \phi^\frown bwd). \end{array}$$

Again, we refer to Section 6.3 for a discussion of how this works when actually using the logics.

### 4.2.3 Other Interval Logics

In this section we consider interval logics that have less in common with SIL than ITL and NL have. As an example, the interval logics discussed so far have incorporated a symbol ($\ell$) giving the length of an interval. This is not the case for the interval logics of this section.

The section is divided into parts where each part is concerned with the work of a specific author (or authors).

#### Allen

In [All84] Allen (who is working in the area of artificial intelligence) proposes a logic where the basic entities simply are "intervals" that are not further specified. He introduces thirteen binary relations on such intervals corresponding to the thirteen possible relationships between intervals [All83]. A logical system is then built, equipped with axioms such as the following:

$$\begin{array}{l} \forall i_1, i_2, i_3(\mathrm{MEETS}(i_1, i_2) \wedge \mathrm{DURING}(i_2, i_3) \rightarrow \\ \qquad (\mathrm{OVERLAPS}(i_1, i_3) \vee \mathrm{DURING}(i_1, i_3) \vee \mathrm{MEETS}(i_1, i_3))), \end{array}$$

where MEETS, OVERLAPS and DURING are three of the 13 binary relations on intervals.

Notice that the above formula is written in pure first order logic. This is the case for the whole logical system which can thus not be seen as an example of a modal interval logic.

We will not consider other non-modal interval logics in this thesis. This despite the fact that other proposals exist if we regard interval logic in a more broad sense.

#### Halpern and Shoham

Some of the first work on interval logic in a more general modal logic setting was that of Halpern and Shoham in [HS91]. They only make a few assumptions concerning ontology and temporal structure: Intervals are built from time points that are totally ordered.

They introduce six unary modalities: $\langle A\rangle$, $\langle\overline{A}\rangle$, $\langle B\rangle$, $\langle\overline{B}\rangle$, $\langle E\rangle$, and $\langle\overline{E}\rangle$. The reading of $\langle A\rangle$ is that $\langle A\rangle\phi$ holds on the current interval iff $\phi$ holds on an interval just to the right of it. The remaining five modalities have the following reading:

- $\langle \overline{A} \rangle \phi$: $\phi$ holds on an interval just to the left of the current interval.
- $\langle B \rangle \phi$: $\phi$ holds on an interval beginning in the same point as the current interval and lying completely within it.
- $\langle \overline{B} \rangle \phi$: $\phi$ holds on an interval beginning in the same point as the current interval.
- $\langle E \rangle \phi$: $\phi$ holds on an interval ending in the same point as the current interval and lying completely within it.
- $\langle \overline{E} \rangle \phi$: $\phi$ holds on an interval ending in the same point as the current interval.

It is shown that all thirteen possible relative positions of two intervals [All83] can be expressed by combining these six modalities appropriately.

Furthermore, Halpern and Shoham discuss how formulas in the logic itself can express certain constraints (such as discreteness or density) on the underlying temporal structure. Finally, the complexity of determining the validity of a formula in the logic is also considered. Except for very simple classes of temporal structures, validity is undecidable.

### Venema

In [Ven90] Venema further develops the work of Halpern and Shoham [HS91]. Firstly, Venema notices that both $\langle A \rangle$ and $\langle \overline{A} \rangle$ are definable in terms of $\langle B \rangle$, $\langle \overline{B} \rangle$, $\langle E \rangle$, and $\langle \overline{E} \rangle$. Then he introduces an intuitive way of thinking of the latter four modalities: The pair of values making up an interval $[i, j]$ can be seen as representing a point in the plane. As $i \leq j$, only points to the left of/above the line $y = x$ can be represented. This will be called the north-western (NW) half plane. Four *compass* modalities $\diamondsuit\!\!\!\!\uparrow, \diamondsuit\!\!\!\!\uparrow, \diamondsuit\!\!\!\!\leftarrow, \diamondsuit\!\!\!\!\rightarrow$ are now introduced by the following definitions:

$$\diamondsuit\!\!\!\!\uparrow \phi \mathrel{\widehat{=}} \langle \overline{B} \rangle \phi,$$
$$\diamondsuit\!\!\!\!\uparrow \phi \mathrel{\widehat{=}} \langle B \rangle \phi,$$
$$\diamondsuit\!\!\!\!\rightarrow \phi \mathrel{\widehat{=}} \langle E \rangle \phi,$$
$$\diamondsuit\!\!\!\!\leftarrow \phi \mathrel{\widehat{=}} \langle \overline{E} \rangle \phi.$$

The intuition is as follows: $\diamondsuit\!\!\!\!\uparrow \phi$ holds at a point $[i, j]$ if there is a point *north* of $[i, j]$ (i.e., a point $[i, k]$ for some $k > j$) where $\phi$ holds. Similarly for the other three modalities.

Utilizing this intuition Venema now proceeds to consider the expressiveness of the logic. He e.g. shows that it is *not* possible to define a binary chop modality $\mathsf{C}$ as an abbreviation in terms of the four compass modalities, where the semantics of $\mathsf{C}$ is as that of $\frown$ of ITL.

In [Ven90] Venema also gives a complete axiomatization for the logic with respect to various temporal structures. Unfortunately, some parts of the proof system are rather complicated.

The work of Venema in [Ven91] builds on the ideas in [HS91] and [Ven90] but now the unary modalities are replaced with three binary modalities $\mathsf{C}$, $\mathsf{T}$, and $\mathsf{D}$.

The modality C is exactly the same as the chop modality mentioned above. The semantics of D and T are as follows: $\phi \mathsf{T} \psi$ holds on $[i,j]$ iff there exists $k \geq j$ such that $\phi$ holds on $[j,k]$ and $\psi$ holds on $[i,k]$; $\phi \mathsf{D} \psi$ holds on $[i,j]$ iff there exists $k \leq i$ such that $\phi$ holds on $[k,i]$ and $\psi$ holds on $[k,j]$.

Venema shows that the four compass modalities of [Ven90] are expressible in terms of C, T, and D, i.e., the system of [Ven90] can be seen as a sublogic of this logic. Finally, Venema describes a complete axiomatization of the system where the proof follows the lines of the completeness proof in [Ven90]. This unfortunately again implies a rather complicated proof system.

## 4.3 Duration Calculus

Duration Calculus (DC) is an extension of "pure" interval logic. DC was originally proposed in [ZHR91] and is concerned with the accumulated durations of Boolean valued state functions over temporal intervals, where time is modeled by the real numbers.

A DC extension to ITL is thouroughly considered in [HZ97]. In this section we will first present a DC extension to SIL, giving Signed Duration Calculus (SDC), and then briefly consider DC extensions to ITL and NL.

### 4.3.1 Signed Duration Calculus

#### Syntax and Semantics

The syntax of SDC is an extension of that of SIL with the introduction of constants with a (possible) special structure: [5]

$$\int S,$$

where $S$ is a *state expression*. State expressions are constructed from a set of *state variables* $P, Q, R, \ldots$ by the following abstract syntax:

$$S \quad ::= \quad 0 \mid 1 \mid P \mid \neg S' \mid S_1 \vee S_2.$$

We will use the standard abbreviations as known from PL. The operators $\neg$ and $\vee$ occur both in state expressions and formulas but they will have different semantics. As state expressions only occur in the context of $\int$ this should not give rise to any confusion.

As mentioned above, DC extensions are concerned with the accumulated durations of Boolean states over temporal intervals, where time is modeled by the real numbers. We thus here only define the semantics of SDC with respect to the class of signed interval models where $T = \mathbb{R}$.

[5] Note that a constant does not *have* to have this structure.

The semantics of terms and formulas defined for SIL is unchanged for SDC except in the cases where a constant has the form $\int S$. To define the semantics of constants in these cases we need an interpretation function $J$ for state variables:

$$J(P) \in \mathbb{R} \to \{0, 1\}.$$

To be sure that the integral of each function $J(P)$ exists on any bounded interval, we must assume that $J(P)$ is finitely variable, i.e., $J(P)$ contains at most a finite number of discontinuity points in any bounded interval.

The interpretation function $J$ is now lifted to state expressions:

$$\begin{array}{lcl} J(0)(t) & = & 0, \\ J(1)(t) & = & 1, \\ J(\neg S)(t) & = & 1 - J(S)(t), \\ J(S_1 \vee S_2)(t) & = & \left\{ \begin{array}{ll} 0 & \text{if } J(S_1)(t) = 0 \text{ and } J(S_2)(t) = 0, \\ 1 & \text{otherwise.} \end{array} \right. \end{array}$$

We see by this definition that since the semantics of state variables are finitely variable functions, the semantics of state expressions will also be finitely variable functions. Hence, these functions contain at most a finite number of discontinuity points in any bounded interval and are thus integrable.

We now redefine the semantics of a constant given an interpretation function $J$:

$$I_J(a)(i,j) = \left\{ \begin{array}{ll} \int_i^j J(S)(\tau) d\tau & \text{if } a \text{ has the form } \int S, \\ I(a)(i,j) & \text{otherwise.} \end{array} \right.$$

Intuitively, the absolute value of the integral can be regarded as the accumulated duration and the sign then just indicates the direction of the interval, in the same way as $\ell$ of SIL gives a signed length.

We introduce the following useful abbreviation (only applicable for SDC formulas):

$$\lceil S \rceil \quad \widehat{=} \quad \int S = \ell \wedge \ell \neq 0 \ , \text{ reads: "}S \text{ high"}.$$

Notice, that by this definition the truth-value of $\lceil S \rceil$ on a signed interval is independent of the direction of the interval, hence

$$\mathfrak{M}_J, \mathcal{V}, (i,j) \models \lceil S \rceil \quad \text{iff} \quad \mathfrak{M}_J, \mathcal{V}, (j,i) \models \lceil S \rceil.$$

This can alternatively be expressed by means of the converse modality:

$$\mathfrak{M}_J, \mathcal{V}, (i,j) \models \lceil S \rceil \leftrightarrow \lceil S \rceil^{-1}.$$

**Proof System**

The proof system for SDC is an extension of that for SIL in the sense that we add axioms to SIL to reflect the (possible) special structure of constants:

DA1: $\int 0 = 0$,

DA2: $\int 1 = \ell$,

DA3: $\ell \geq 0 \rightarrow \int S \geq 0$,
$\ell \leq 0 \rightarrow \int S \leq 0$,

DA4: $\int S_1 + \int S_2 = \int (S_1 \vee S_2) + \int (S_1 \wedge S_2)$,

DA5: $(\int S = s)^\frown(\int S = t) \rightarrow (\int S = s + t)$ if $s$ and $t$ are rigid,

DA6: $\int S_1 = \int S_2$ if $S_1 \leftrightarrow S_2$ in propositional logic.

Furthermore, we adjoin the following induction rules:

$$\frac{(\ell = 0) \rightarrow \phi \quad (\phi \ \barwedge \ \lceil S \rceil) \rightarrow \phi \quad (\phi \ \barwedge \ \lceil \neg S \rceil) \rightarrow \phi}{\phi} \text{ DRr}$$

$$\frac{(\ell = 0) \rightarrow \phi \quad (\lceil S \rceil \ \barwedge \ \phi) \rightarrow \phi \quad (\lceil \neg S \rceil \ \barwedge \ \phi) \rightarrow \phi}{\phi} \text{ DRl} \ .$$

Note that the contracting chop $\barwedge$ is used here; the induction rules are not sound for the general $^\frown$ [Ras99b]. DRr and DRl are actually stronger than necessary for relative completeness, cf. Section 4.3.3 below. In [Ras99b], more complicated (but still sound) rules are considered as well (from which the above can be derived). For our purposes in this thesis, DRr and DRl are sufficient, and we will therefore not consider such more complicated rules.

### 4.3.2 DC for ITL and NL

Here we briefly sketch the principles of DC extensions to ITL and NL.

In the case of ITL the syntax and semantics defined for SDC above is carried over completely unchanged — the only difference appears when considering the underlying SIL and ITL as such. In particular, the fact that intervals of ITL all are forward makes sure that the integral in the semantics always will be positive.

This is reflected in the proof system where the axiom DA3 is changed to:

DA3': $\int S \geq 0$.

With respect to the induction rules, more complicated versions than those of SDC above are usually chosen, cf., e.g., [HZ97].

In NL, the chop of ITL can be defined as an abbreviated modality. Based on this, a DC extension can now be defined in the exact same way as that for ITL. The only exception is in the case of the induction rules where slight modifications are necessary to retain soundness. Details are given in [RZ97].

### 4.3.3 Soundness and (Relative) Completeness

The DC extension to SIL is sound. This is fairly simple to show in case of the axioms. In the case of the induction rules, the situation is a little more complicated; a proof is given in [Ras99b].

As the underlying interval logic of SDC is based on the concrete temporal domain $\mathbb{R}$ we can not hope for completeness of the full system. What we instead aim for is *relative* completeness, by which is meant that if every valid SIL formula is taken as a theorem then any valid SDC formula can be proved. In [Ras99b] it is argued that this is the case for SDC. The argument is based on a similar argument for DC [HZ97].

For such relative completeness, induction rules (as discussed above) are actually not necessary. All that is needed are the following simple axioms:

$$(\ell = 0) \vee (\text{true} \;\; \frown \;\; \lceil S \rceil) \vee (\text{true} \;\; \frown \;\; \lceil \neg S \rceil),$$
$$(\ell = 0) \vee (\lceil S \rceil \;\; \frown \;\; \text{true}) \vee (\lceil \neg S \rceil \;\; \frown \;\; \text{true}).$$

A more detailed discussion on relative completeness of SDC can be found in [Ras99b].

We end this section by asking whether a kind of DC extension would be possible for an interval logic based on abstract domains and not $\mathbb{R}$? At first sight, this would require generalizing integration to such abstract domains (which is not a trivial task, to say the least). But one notices that the kind of integration being carried out in the case of DC is quite restricted due to the fact that state formulas are Boolean and finitely variable. Taking this into account, a completeness result for DC on abstract domains is given [Gue98].

CHAPTER 5

# Sequent Calculus

In the previous chapter we gave an introduction to interval logic with emphasis on SIL. All proof systems of that chapter were formulated as Hilbert systems. This was convenient for establishing meta-results such as the important completeness results. But our goal is now to investigate and improve formalisms for actual proof-making.

In this chapter we consider a sequent calculus proof system for SIL. In Section 5.1 we begin by presenting the rules making up the system, we more closely consider the structure of the rules and we show how a weakened version of the subformula property can be achieved. We then sketch a proof of the essential result that the sequent calculus system is equivalent to the Hilbert system. In Section 5.2 we investigate our proposed sequent calculus with respect to decidability/undecidability. We show that the limit between decidability and undecidability of quantifier-free SIL is the cut-rule (Cut). We end the chapter, in Section 5.3, by concluding from both a theoretical and a pragmatic viewpoint.

## 5.1 The Sequent Rules

In this section we present a number of sequent rules which together make up our sequent calculus for SIL.

SIL is an extension of FOL. We thus include the rules L (for PL) (cf. Section 2.1.4) and the rules P (for FOL) (cf. Section 2.2.4). But as was the case for the Hilbert system (cf. Section 4.1.2) we have to modify the quantifier rules slightly. Let P′ denote the set of the following four sequent rules:

$$\frac{\Gamma, \phi[s/x] \vdash \Delta}{\Gamma, (\forall x)\phi \vdash \Delta}\ (\mathrm{L}\forall) \qquad \frac{\Gamma \vdash \phi, \Delta}{\Gamma \vdash (\forall x)\phi, \Delta}\ (\mathrm{R}\forall)$$

$$\frac{\Gamma, \phi \vdash \Delta}{\Gamma, (\exists x)\phi \vdash \Delta}\ (\mathrm{L}\exists) \qquad \frac{\Gamma \vdash \phi[s/x], \Delta}{\Gamma \vdash (\exists x)\phi, \Delta}\ (\mathrm{R}\exists)\ ,$$

where the following sideconditions apply:

- (L$\exists$) and (R$\forall$): $x \notin \mathrm{FV}(\Gamma \cup \Delta)$.
- (L$\forall$) and (R$\exists$): $s$ is rigid or $\phi$ is chop-free.

We also include the rules E for equality (cf. Section 2.2.4). If the only predicate symbol is $=$ and the only non-nullary function symbols are $+$ and $-$ we denote the equality rules E$'$.

The only interval modality of SIL is $^\frown$ and we of course also need rules defining the properties of this connective. However, it turns out convenient to base the system on the sequent rules for S4 discussed in Section 3.4. We can do this because the two unary modalities $\Box$ and $\Diamond$ are definable in SIL as follows:

$$\Diamond\phi \mathrel{\widehat{=}} \mathrm{true}^\frown\phi^\frown\mathrm{true},$$
$$\Box\phi \mathrel{\widehat{=}} \neg\Diamond\neg\phi.$$

In this setting, $\Diamond\phi$ can be read as "for some signed interval $\phi$" and $\Box\phi$ as "for all signed intervals $\phi$". It is easy to formally show that these definitions imply the properties of S4 listed in Section 3.3. Thus, we include the rules denoted 4 in Section 3.4 for S4, viz.

$$\frac{\Gamma, \phi \vdash \Delta}{\Gamma, \Box\phi \vdash \Delta}\ (\mathrm{L}\Box) \qquad \frac{\Box\Gamma \vdash \phi, \Diamond\Delta}{\Gamma', \Box\Gamma \vdash \Box\phi, \Diamond\Delta, \Delta'}\ (\mathrm{R}\Box'')$$

$$\frac{\Box\Gamma, \phi \vdash \Diamond\Delta}{\Gamma', \Box\Gamma, \Diamond\phi \vdash \Diamond\Delta, \Delta'}\ (\mathrm{L}\Diamond'') \qquad \frac{\Gamma \vdash \phi, \Delta}{\Gamma \vdash \Diamond\phi, \Delta}\ (\mathrm{R}\Diamond)\ ,$$

where $\Box$ and $\Diamond$ are abbreviations as defined above.

When adding the following rules we collectively refer to the S4-rules as $4'$:

$$\frac{\Gamma, \phi \vdash \Delta}{\Gamma, \Diamond\phi \vdash \Delta}\ (\mathrm{LR}) \qquad \frac{\Gamma \vdash \phi, \Delta}{\Gamma \vdash \Box\phi, \Delta}\ (\mathrm{RR})\ ,$$

where the following sidecondition apply for both rules:

- $\phi$ is rigid.

These rules are (essentially) (L$\Diamond''$) and (R$\Box''$), respectively, but with somewhat different sideconditions.

We now turn to more explicit rules for $\frown$. We will denote the following sequent rules by I:

$$\frac{\Gamma, (\ell = s + t) \vdash \Delta}{\Gamma, (\ell = s)\frown(\ell = t) \vdash \Delta}\ (\text{LL2}) \qquad \frac{\Gamma \vdash (\ell = s + t), \Delta}{\Gamma \vdash (\ell = s)\frown(\ell = t), \Delta}\ (\text{RL2})$$

$$\frac{\Gamma, \phi \vdash \Delta}{\Gamma, \phi\frown(\ell = 0) \vdash \Delta}\ (\text{LL3}) \qquad \frac{\Gamma \vdash \phi, \Delta}{\Gamma \vdash \phi\frown(\ell = 0), \Delta}\ (\text{RL3})$$

$$\frac{\Gamma, \phi \vdash \Delta}{\Gamma, (\ell = 0)\frown\phi \vdash \Delta}\ (\text{LL3}) \qquad \frac{\Gamma \vdash \phi, \Delta}{\Gamma \vdash (\ell = 0)\frown\phi, \Delta}\ (\text{RL3})$$

$$\frac{\Gamma, \phi\frown\chi \vdash \Delta \quad \Gamma, \psi\frown\chi \vdash \Delta}{\Gamma, (\phi \vee \psi)\frown\chi \vdash \Delta}\ (\text{LT1}) \qquad \frac{\Gamma \vdash \phi\frown\chi, \psi\frown\chi, \Delta}{\Gamma \vdash (\phi \vee \psi)\frown\chi, \Delta}\ (\text{RT1})$$

$$\frac{\Gamma, \chi\frown\phi \vdash \Delta \quad \Gamma, \chi\frown\psi \vdash \Delta}{\Gamma, \chi\frown(\phi \vee \psi) \vdash \Delta}\ (\text{LT1}) \qquad \frac{\Gamma \vdash \chi\frown\phi, \chi\frown\psi, \Delta}{\Gamma \vdash \chi\frown(\phi \vee \psi), \Delta}\ (\text{RT1})$$

$$\frac{\Gamma \vdash (\ell = s)\frown\phi, \Delta}{\Gamma, (\ell = s)\frown\neg\phi \vdash \Delta}\ (\text{LL1}) \qquad \frac{\Gamma, (\ell = s)\frown\phi \vdash \Delta}{\Gamma \vdash (\ell = s)\frown\neg\phi, \Delta}\ (\text{RL1})$$

$$\frac{\Gamma \vdash \phi\frown(\ell = s), \Delta}{\Gamma, \neg\phi\frown(\ell = s) \vdash \Delta}\ (\text{LL1}) \qquad \frac{\Gamma, \phi\frown(\ell = s) \vdash \Delta}{\Gamma \vdash \neg\phi\frown(\ell = s), \Delta}\ (\text{RL1})$$

$$\frac{\Box\Gamma, \phi \vdash \psi, \Diamond\Delta}{\Gamma', \Box\Gamma, \phi\frown\chi \vdash \psi\frown\chi, \Diamond\Delta, \Delta'}\ (\text{LRM})$$

$$\frac{\Box\Gamma, \phi \vdash \psi, \Diamond\Delta}{\Gamma', \Box\Gamma, \chi\frown\phi \vdash \chi\frown\psi, \Diamond\Delta, \Delta'}\ (\text{LRM}),$$

where the following sideconditions apply:

- (LL2) and (RL2): $s$ and $t$ are rigid.
- (LL1) and (RL1): $s$ is rigid.

Note that rules only distinguished by symmetric formulas with respect to $\frown$ are given the same name. The rules (LL3), (RT1), (RL1), (RBl) and (RBr) (see below for the latter two) are in fact redundant in the sense that they are derivable given the other rules. We include them explicitly as it results in a more symmetric system.

The following rules, expressing associativity of $\frown$, will be denoted A:

$$\frac{\Gamma, \phi\frown(\chi\frown\psi) \vdash \Delta}{\Gamma, (\phi\frown\chi)\frown\psi \vdash \Delta}\ (\text{LA2}) \qquad \frac{\Gamma \vdash \phi\frown(\chi\frown\psi), \Delta}{\Gamma \vdash (\phi\frown\chi)\frown\psi, \Delta}\ (\text{RA2}).$$

We also need rules expressing the interplay between the quantifiers and $^\frown$. Let Q denote the following rules:

$$\frac{\Gamma, \phi^\frown\psi \vdash \Delta}{\Gamma, ((\exists x)\phi)^\frown\psi \vdash \Delta}\ (\text{LBl}) \qquad \frac{\Gamma \vdash \phi[s/x]^\frown\psi, \Delta}{\Gamma \vdash ((\exists x)\phi)^\frown\psi\Delta}\ (\text{RBl})$$

$$\frac{\Gamma, \phi^\frown\psi \vdash \Delta}{\Gamma, \phi^\frown((\exists x)\psi) \vdash \Delta}\ (\text{LBr}) \qquad \frac{\Gamma \vdash \phi^\frown\psi[s/x], \Delta}{\Gamma \vdash \phi^\frown((\exists x)\psi), \Delta}\ (\text{RBr}) \ ,$$

where the following sideconditions apply

- (LBl): $x \notin \mathrm{FV}(\Gamma \cup \Delta \cup \{\psi\})$.
- (LBr): $x \notin \mathrm{FV}(\Gamma \cup \Delta \cup \{\phi\})$.
- (RBl): $s$ is rigid or $\phi$ is chop-free.
- (RBr): $s$ is rigid or $\psi$ is chop-free.

Finally, we need axioms expressing the properties of an Abelian group. Denote the following rules by G:

$$\frac{}{\Gamma \vdash (s+t)+u = s+(t+u), \Delta}\ (\text{SD1}) \qquad \frac{}{\Gamma \vdash s+0 = s, \Delta}\ (\text{SD2})$$

$$\frac{}{\Gamma \vdash s+(-s) = 0, \Delta}\ (\text{SD3}) \qquad \frac{}{\Gamma \vdash s+t = t+s, \Delta}\ (\text{SD4}) \ .$$

We have now presented all logical sequent rules for the SIL system. Of structural rules, only the cut rule (Cut) will be included.

As discussed in Section 2.1.4, the exchange rules are superfluous when we consider sequents of multisets. The weakening rules are absorbed in the logical rules; this is already the case for the two S4 rules (R$\Box''$) and (L$\Diamond''$) (cf. Section 3.4). Of the new rules of this chapter, (LRM) has a form which shows that the weakening rules have been absorbed. Finally, the contraction rules are easily derivable when cut is included, hence they are not included explicitly.

It is *not* possible to eliminate (Cut) from the system. This is a corollary of the undecidability/decidability result of Section 5.2 as we shall see (Corollary 5.16).

We can now be precise:

**Definition 5.1** The sequent calculus for SIL is $\mathfrak{G}[\mathrm{LP}'\mathrm{E4}'\mathrm{IAQG}, (\mathrm{Cut})]$ .

To ease readability we will write $\mathfrak{G}_{\mathrm{SIL}}$ for $\mathfrak{G}[\mathrm{LP}'\mathrm{E4}'\mathrm{IAQG}, (\mathrm{Cut})]$ and $\vdash_{\mathfrak{G}_{\mathrm{SIL}}} \phi$ for $\vdash_{\mathfrak{G}[\mathrm{LP}'\mathrm{E4}'\mathrm{IAQG},(\mathrm{Cut})]} \phi$.

In Section 5.1.2 we will show that $\mathfrak{G}_{\mathrm{SIL}}$ is equivalent to the Hilbert system for SIL. But first, in the following section, we discuss the structure of the rules presented so far and consider some consequences hereof.

### 5.1.1 Structure of the Rules

The rules of LP4' are all well-known and have been discussed earlier. In particular, note/recall that they all satisfy the standard subformula property (Definition 2.9).

The rules (LL2), (RL2), (LL3) and (RL3) express how interval lengths are additive and that an interval of length zero is a neutral element with respect to $\frown$. The rules (LL3) and (RL3) satisfy the subformula property and as the formulas in the premises of (LL2) and (RL2) are atomic, all four rules are suited for backwards proof search (cf. the very last part of Section 2.1.4). The rules (LT1), (RT1), (LL1), (RL1), (LBl), (LBr), (RBl) and (RBr) all have a particular form: They can be seen as introduction rules for $\vee$, $\neg$ and $\exists$ "under the chop". In other words, these rules resemble the corresponding rules for propositional logic but now the affected formulas are chopped formulas. (It is possible to derive similar rules for $\wedge$, $\rightarrow$ and $\forall$.) Note that these rules do not satisfy the usual subformula property. But because of their particular form it is possible to define a weakened subformula property which gives rise to a decreasing measure in a backwards proof search. This is the case for the monotonicity rules for $\frown$ (LRM) too.

**Definition 5.2** We say that $\phi$ is a *chop-subformula* of $\psi$ if one of the following propositions holds:

1. $\phi$ is a subformula of $\psi$,
2. $\phi$ has the form $\phi_1 \frown \phi_2$, $\psi$ has the form $\psi_1 \frown \psi_2$, and $\phi_1$ is a subformula of $\psi_1$ and $\phi_2$ is a subformula of $\psi_2$,
3. $\phi$ is atomic and $\psi$ is not atomic.

Furthermore, $\phi$ is a *proper chop-subformula* of $\psi$ if $\phi$ is a chop-subformula of $\psi$ and $\phi$ is not $\psi$.

An example: $\phi \frown \chi$ is a chop-subformula of $(\phi \vee \psi) \frown \chi$.

**Definition 5.3** A sequent rule has the *chop-subformula property* if for each active formula $\phi$ there exists a principal formula $\psi$ such that $\phi$ is a proper chop-subformula of $\psi$.

This definition corresponds to Definition 2.9. We see that the rules of L, $4'$ and I all satisfy the chop-subformula property.

As in Section 2.1.4 we now want to associate a measure with a sequent. This time the measure should decrease when rules satisfying the chop-subformula property are used in a backwards proof search. But this time we have to be a little more clever than just taking the sum of the sizes, as this would not give the desired result — consider, e.g., (RT1).

**Definition 5.4** Let $\nu(\phi)$ denote the size of $\phi$. This extends to multisets, $\nu(\Gamma) = \{\nu(\phi) \mid \phi \in \Gamma\}$, and to sequents, $\nu(\Gamma \vdash \Delta) = \nu(\Gamma) \cup \nu(\Delta)$.

Let $\mu(\phi)$ be defined as follows:

$$\mu(\phi) = \begin{cases} 1 + \mu(\phi_1) & \text{if } \phi \text{ is } \phi_1{}^\frown\phi_2 \text{ for some } \phi_1, \phi_2, \\ 0 & \text{otherwise.} \end{cases}$$

This extends to multisets and sequents as well.

Both $\nu$ and $\mu$ give us multisets of natural numbers when applied to sequents. Such multisets can be well-ordered by a *multiset ordering* [BN98]. Now, $\nu$ of a premise of a rule satisfying the chop-subformula property will be less than $\nu$ of the conclusion with respect to such an ordering. Similarly, $\mu$ of the premise of a rule of A will be less than $\mu$ of the conclusion. We can now define an order relation $\prec$ on sequents as the lexicographic product of the multiset orderings on sequents as given by $\nu$ and $\mu$, respectively.

In conclusion, a backwards proof search performed on a sequent using the rules of L, $4'$, I and A will terminate as $\prec$ will decrease strictly for each step.

### 5.1.2 Equivalence

In this section we show that theoremhood in the Hilbert system and theoremhood in the sequent calculus system coincide.

**Lemma 5.5** *All (Hilbert system) axioms of SIL are provable in* $\mathfrak{G}_{\mathrm{SIL}}$. *All (Hilbert system) inference rules of SIL are derivable in* $\mathfrak{G}_{\mathrm{SIL}}$.

Proof. The proof is straightforward. The parts concerning FOLE are standard. For the SIL extension we only give a couple of examples.

First we show that the axiom L1 is provable in $\mathfrak{G}_{\mathrm{SIL}}$:

$$\dfrac{\dfrac{\dfrac{(\ell = s)^\frown\phi \;\vdash\; (\ell = s)^\frown\phi}{(\ell = s)^\frown\phi, (\ell = s)^\frown\neg\phi \;\vdash}\,(\mathrm{LL1})}{(\ell = s)^\frown\phi \;\vdash\; \neg((\ell = s)^\frown\neg\phi)}\,(\mathrm{R}\neg)}{\vdash\; (\ell = s)^\frown\phi \to \neg((\ell = s)^\frown\neg\phi)}\,(\mathrm{R}\to)\ .$$

We now show that the rule M is derivable in $\mathfrak{G}_{\mathrm{SIL}}$. In other words, we show that if we assume $\vdash\ \phi \to \psi$ then we can prove $\vdash\ \phi^\frown\chi \to \psi^\frown\chi$:

$$\dfrac{\dfrac{\dfrac{\vdash\ \phi \to \psi}{\phi \;\vdash\; \psi, \phi \to \psi}\,(\mathrm{LW}),(\mathrm{RW}) \qquad \dfrac{\phi \;\vdash\; \phi, \psi \quad \phi, \psi \;\vdash\; \psi}{\phi, \phi \to \psi \;\vdash\; \psi}\,(\mathrm{L}\to)}{\phi \;\vdash\; \psi}\,(\mathrm{Cut})}{\phi^\frown\chi \;\vdash\; \psi^\frown\chi}\,(\mathrm{LRM})}{\vdash\; \phi^\frown\chi \to \psi^\frown\chi}\,(\mathrm{R}\to)\ .$$

□

**Definition 5.6** If $\Gamma = \{\phi_1, \phi_2, \ldots, \phi_n\}$ is a multiset of formulas then

$$\begin{array}{rcl} \bigwedge \Gamma & \widehat{=} & \phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_n, \\ \bigvee \Gamma & \widehat{=} & \phi_1 \vee \phi_2 \vee \cdots \vee \phi_n. \end{array}$$

In particular, if $n = 0$ then $\Gamma = \emptyset$, and $\bigwedge \Gamma \widehat{=}$ true and $\bigvee \Gamma \widehat{=}$ false.

**Lemma 5.7** *The following are theorems of SIL:*

1. $(\bigwedge \Box\Gamma) \leftrightarrow (\Box \bigwedge \Gamma)$.
2. $(\bigvee \Diamond\Gamma) \leftrightarrow (\Diamond \bigvee \Gamma)$.
3. $\Box(\phi \vee \Diamond\psi) \rightarrow \Box\phi \vee \Diamond\psi$.
4. $(\Diamond\phi)^\frown\psi \rightarrow \Diamond\phi$.
5. *If* $\Box\phi \rightarrow \psi$ *then* $\Box\phi \rightarrow \Box\psi$ .

**Lemma 5.8** *For all sequent rules*

$$\frac{\Gamma_1 \vdash \Delta_1 \quad \Gamma_2 \vdash \Delta_2 \quad \cdots \quad \Gamma_n \vdash \Delta_n}{\Gamma' \vdash \Delta'} \ ,$$

*of* $\mathfrak{G}_{\mathrm{SIL}}$,

$$\textit{if} \quad \left\{ \begin{array}{c} \vdash_{\mathrm{SIL}} \bigwedge \Gamma_1 \rightarrow \bigvee \Delta_1 \\ \vdash_{\mathrm{SIL}} \bigwedge \Gamma_2 \rightarrow \bigvee \Delta_2 \\ \vdots \\ \vdash_{\mathrm{SIL}} \bigwedge \Gamma_n \rightarrow \bigvee \Delta_n \end{array} \right\} \quad \textit{then} \quad \vdash_{\mathrm{SIL}} \bigwedge \Gamma' \rightarrow \bigvee \Delta'.$$

Proof. The proof is not complicated but can be tedious. We just give a few examples showing how the reasoning goes.

When we in the Hilbert proofs below write 'PL' we refer to simple propositional reasoning. Otherwise we refer to explicit axioms/rules.

Case (R$\Box''$): We assume that $\vdash_{\mathrm{SIL}} \bigwedge \Box\Gamma \rightarrow \phi \vee \bigvee \Diamond\Delta$. We must then show that $\vdash_{\mathrm{SIL}} \bigwedge \Gamma' \wedge \bigwedge \Box\Gamma \rightarrow \Box\phi \vee \bigvee \Diamond\Delta \vee \bigvee \Diamond\Delta'$:

| | | |
|---|---|---|
| 1. | $\bigwedge \Box\Gamma \rightarrow \phi \vee \bigvee \Diamond\Delta$ | Assumption. |
| 2. | $\Box \bigwedge \Gamma \rightarrow \phi \vee \Diamond \bigvee \Delta$ | Lemma 5.7 (*1.* and *2.*), PL, 1. |
| 3. | $\Box \bigwedge \Gamma \rightarrow \Box(\phi \vee \Diamond \bigvee \Delta)$ | Lemma 5.7 (*5.*), 2. |
| 4. | $\Box \bigwedge \Gamma \rightarrow \Box\phi \vee \Diamond \bigvee \Delta$ | Lemma 5.7 (*3.*), PL, 3. |
| 5. | $\bigwedge \Box\Gamma \rightarrow \Box\phi \vee \bigvee \Diamond\Delta$ | Lemma 5.7 (*1.* and *2.*), PL, 4. |
| 6. | $\bigwedge \Gamma' \wedge \bigwedge \Box\Gamma \rightarrow \Box\phi \vee \bigvee \Diamond\Delta \vee \bigvee \Diamond\Delta'$ | PL, 5. |

Case (LBl): Assume $\vdash_{\mathrm{SIL}} \bigwedge\Gamma \wedge \phi^\frown\psi \rightarrow \bigvee\Delta$. We must then show that when $x \notin \mathrm{FV}(\Gamma \cup \Delta \cup \{\psi\})$, $\vdash_{\mathrm{SIL}} \bigwedge\Gamma \wedge ((\exists x)\phi)^\frown\psi \rightarrow \bigvee\Delta$:

| | | |
|---|---|---|
| 1. | $\bigwedge\Gamma \wedge \phi^\frown\psi \rightarrow \bigvee\Delta$ | Assumption. |
| 2. | $\bigwedge\Gamma \wedge \neg(\bigvee\Delta) \rightarrow \neg(\phi^\frown\psi)$ | PL, 1. |
| 3. | $(\forall x)(\bigwedge\Gamma \wedge \neg(\bigvee\Delta) \rightarrow \neg(\phi^\frown\psi))$ | G, 2. |
| 4. | $\bigwedge\Gamma \wedge \neg(\bigvee\Delta) \rightarrow (\forall x)\neg(\phi^\frown\psi)$ | Q2, PL, 3. |
| 5. | $\bigwedge\Gamma \wedge (\exists x)(\phi^\frown\psi) \rightarrow \bigvee\Delta$ | PL, 4. |
| 6. | $\bigwedge\Gamma \wedge ((\exists x)\phi)^\frown\psi \rightarrow \bigwedge\Gamma \wedge (\exists x)(\phi^\frown\psi)$ | B, PL. |
| 7. | $\bigwedge\Gamma \wedge ((\exists x)\phi)^\frown\psi \rightarrow \bigvee\Delta$ | PL, 5., 6. |

Case (LRM): For this case we assume $\vdash_{\mathrm{SIL}} \bigwedge\Box\Gamma \wedge \phi \rightarrow \psi \vee \bigvee\Diamond\Delta$. We must then show $\vdash_{\mathrm{SIL}} \bigwedge\Gamma' \wedge \bigwedge\Box\Gamma \wedge \phi^\frown\chi \rightarrow \psi^\frown\chi \vee \bigvee\Diamond\Delta \vee \bigvee\Diamond\Delta'$:

| | | |
|---|---|---|
| 1. | $\bigwedge\Box\Gamma \wedge \phi \rightarrow \psi \vee \bigvee\Diamond\Delta$ | Assumption |
| 2. | $\Box\bigwedge\Gamma \wedge \phi \rightarrow \psi \vee \Diamond\bigvee\Delta$ | Lemma 5.7 (*1.* and *2.*), PL, 1. |
| 3. | $\phi \rightarrow \psi \vee \Diamond\neg\bigwedge\Gamma \vee \Diamond\bigvee\Delta$ | Def. $\Box$, PL, 2. |
| 4. | $\phi^\frown\chi \rightarrow \psi^\frown\chi \vee (\Diamond\neg\bigwedge\Gamma)^\frown\chi \vee (\Diamond\bigvee\Delta)^\frown\chi$ | M, K, PL, 3. |
| 5. | $\phi^\frown\chi \rightarrow \psi^\frown\chi \vee \Diamond\neg\bigwedge\Gamma \vee \Diamond\bigvee\Delta$ | Lemma 5.7 (*4.*), PL, 4. |
| 6. | $\Box\bigwedge\Gamma \wedge \phi^\frown\chi \rightarrow \psi^\frown\chi \vee \Diamond\bigvee\Delta$ | Def. $\Box$, PL, 5. |
| 7. | $\bigwedge\Box\Gamma \wedge \phi^\frown\chi \rightarrow \psi^\frown\chi \vee \bigvee\Diamond\Delta$ | Lemma 5.7 (*1.* and *2.*), PL, 6. |

From 7. we are then done by PL. □

We can now state the central result of this section.

**Theorem 5.9**

$\vdash_{\mathrm{SIL}} \phi$ *iff* $\vdash_{\mathfrak{G}_{\mathrm{SIL}}} \phi$.

Proof. The *only if* direction is proven by induction on the length of the proof of $\phi$ in the Hilbert system. This is straightforward by using Lemma 5.5.

For the *if* direction we prove a stronger result: If there is a proof of $\Gamma \vdash \Delta$ in $\mathfrak{G}_{\mathrm{SIL}}$ then $\vdash_{\mathrm{SIL}} \bigwedge\Gamma \rightarrow \bigvee\Delta$. (This trivially implies the *if*-part.) The proof is by structural induction over the proof tree for $\Gamma \vdash \Delta$: If $\Gamma \vdash \Delta$ is a basic sequent we are done — otherwise we use Lemma 5.8. □

## 5.2 Decidability Modulo Cut

SIL is an extension of FOL — SIL is therefore undecidable because FOL is. In this section we consider Quantifier Free SIL ($\mathrm{SIL}_{\mathrm{QF}}$) with $=$ being the only predicate symbol and $+$ and $-$ being the only non-nullary function symbols. We show that the limit between decidability and undecidability of $\mathrm{SIL}_{\mathrm{QF}}$ is the (Cut) rule.

### 5.2.1 Undecidability

First we show that $\mathrm{SIL}_{\mathrm{QF}}$ is undecidable in general, i.e., we show that it is undecidable whether $\vdash_{\mathrm{SIL}} \phi$ for arbitrary $\phi$ of $\mathrm{SIL}_{\mathrm{QF}}$. But this is clearly the case if it is undecidable whether $\vdash_{\mathrm{SIL}} \alpha$ for arbitrary $\alpha$ of (propositional) $\mathcal{L}^{\frown}$. (Recall that SIL is an extension of (propositional) $\mathcal{L}^{\frown}$ such that a formula of $\mathcal{L}^{\frown}$ will be a formula of $\mathrm{SIL}_{\mathrm{QF}}$ as well.) By the soundness and completeness theorem for SIL (Theorem 4.14) this is equivalent to the decidability question for $\models_{\mathrm{SIL}} \alpha$. Recalling the undecidability result of Section 3.3 (Theorem 3.5) we are done by the following proposition.

**Proposition 5.10**

$$\models_{\mathrm{SIL}} \alpha \quad \textit{iff} \quad \models_{\mathrm{SQ}} \alpha.$$

Proof. By simple structural induction over the definitions of $\models_{\mathrm{SIL}}$ and $\models_{\mathrm{SQ}}$. □

### 5.2.2 A Decidable Fragment

We now show that $\mathrm{SIL}_{\mathrm{QF}}$ without (Cut) is decidable. To be more precise, we show that it is decidable whether a sequent of $\mathrm{SIL}_{\mathrm{QF}}$ is provable in $\mathfrak{G}[\mathrm{LP'E'4'IAQG}]$. First some simple definitions.

An *atomic basic sequent* $\Gamma \vdash \Delta$ is a basic sequent where all formulas of $\Gamma \cap \Delta$ are atomic. A proof in an *atomic sequent calculus* $\overline{\mathfrak{G}}[\mathcal{R}]$ is a proof in $\mathfrak{G}[\mathcal{R}]$ where basic sequents are atomic. Let $\overline{\Gamma} \mathrel{\widehat{=}} \{\phi \in \Gamma \mid \phi \text{ is atomic}\}$.

As formulas of $\mathrm{SIL}_{\mathrm{QF}}$ are quantifier-free, variables can never be instantiated and can thus be regarded as constants. Such formulas with no variables are called *ground*.

If a sequent $\Gamma \vdash \Delta$ is an instance of the conclusion of a sequent rule $R$, we say that $R$ is (backwards) *applicable* to $\Gamma \vdash \Delta$.

The following lemma states that after using the (terminating) backwards proof search discussed in Section 5.1.1, what is left is to use equality reasoning on Abelian groups.

**Lemma 5.11** *Given a non-basic sequent of* $\mathrm{SIL}_{\mathrm{QF}}$*, if none of the sequent rules of* L4′IA *are applicable, then the following propositions are equivalent:*

1. *There is a proof of* $\Gamma \vdash \Delta$ *in* $\mathfrak{G}[\mathrm{LP'E'4'IAQG}]$.
2. *There is a proof of* $\overline{\Gamma} \vdash \overline{\Delta}$ *in* $\overline{\mathfrak{G}}[\mathrm{E'G}]$.

Proof. Trivially, *2.* implies *1.* For the other direction, notice that as we only consider quantifier-free formulas the sequent rules of P′ and Q will never be applicable. We can therefore restrict attention to provability in $\mathfrak{G}[\mathrm{LE'4'IAG}]$. By assumption, none of the sequent rules of L4′IA are applicable. Of the remaining rules, only those of E′ can generate new sequents; but the additional formulas of those will always be atomic, hence the rules of L4′IA will continue being non-applicable. Hence, we only have to consider provability in $\mathfrak{G}[\mathrm{E'G}]$.

Assume $\Gamma \vdash \Delta$ is provable in $\mathfrak{G}[\mathrm{E}'\mathrm{G}]$ because $\Gamma \vdash \Delta$ is an instance of an axiom of G. Then clearly $\overline{\Gamma} \vdash \overline{\Delta}$ will be an instance of the same axiom of G and we have a proof in $\overline{\mathfrak{G}}[\mathrm{E}'\mathrm{G}]$. The only possibility left is that $\Gamma \vdash \Delta$ is provable in $\mathfrak{G}[\mathrm{E}'\mathrm{G}]$ because one of the rules of $\mathrm{E}'$ is applied to $\Gamma \vdash \Delta$. In this case there are three possibilities for each of the new sequents $\Gamma' \vdash \Delta'$:

1. $\Gamma' \vdash \Delta'$ is an instance of an axiom of G. Then we are done as above.
2. $\Gamma' \vdash \Delta'$ is a basic sequent. As $\Gamma \vdash \Delta$ was not a basic sequent $\Gamma' \vdash \Delta'$ must be an atomic basic sequent (because of the structure of the rules of $\mathrm{E}'$). Thus, we have a proof of $\overline{\Gamma'} \vdash \overline{\Delta'}$ in $\overline{\mathfrak{G}}[\mathrm{E}'\mathrm{G}]$.
3. $\Gamma' \vdash \Delta'$ is provable in $\mathfrak{G}[\mathrm{E}'\mathrm{G}]$ because one of the rules of $\mathrm{E}'$ is applied to $\Gamma \vdash \Delta$. Then we are done by induction.

□

An *equational system* $E$ is a set of *equations* $s = t$ where $s$ and $t$ are terms built from function symbols and variables (as in FOL). A *structure* $M$ consists of a domain $D$ and a function assigning a meaning (in $D$) to each function symbol and variable. The meaning of terms is defined by lifting in the usual way. We say that $M$ *satisfies* the equation $s = t$ iff $s$ and $t$ are given the same meaning by $M$. We write $E \models_{\mathrm{equ}} s = t$ if all structures that satisfy all equations in $E$ also satisfy $s = t$.

A *substitution* $\sigma$ is a function from variables to terms which is identity on all but finitely many variables. A substitution can be lifted to terms the obvious way.

We now define the relation $E \vdash_{\mathrm{equ}} s = t$ between equational systems $E$ and equations $s = t$ as the least relation which satisfies the assumption rule:

$$E \vdash_{\mathrm{equ}} s = t \quad \text{if} \quad (s = t) \in E \quad \text{(QA)},$$

and is closed under the following inference rules:

$$\frac{}{E \vdash_{\mathrm{equ}} t = t}\ \text{(QR)} \qquad \frac{E \vdash_{\mathrm{equ}} s = t}{E \vdash_{\mathrm{equ}} t = s}\ \text{(QS)}$$

$$\frac{E \vdash_{\mathrm{equ}} s = t \quad E \vdash_{\mathrm{equ}} t = u}{E \vdash_{\mathrm{equ}} s = u}\ \text{(QT)} \qquad \frac{E \vdash_{\mathrm{equ}} s = t}{E \vdash_{\mathrm{equ}} \sigma(s) = \sigma(t)}\ \text{(QV)}$$

$$\frac{E \vdash_{\mathrm{equ}} s_1 = t_1 \quad \cdots \quad E \vdash_{\mathrm{equ}} s_n = t_n}{E \vdash_{\mathrm{equ}} f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n)}\ \text{(QC)} \ .$$

There is a classical result relating $\models_{\mathrm{equ}}$ and $\vdash_{\mathrm{equ}}$ [Pla93].

**Theorem 5.12**

$$E \models_{\mathrm{equ}} s = t \quad \textit{iff} \quad E \vdash_{\mathrm{equ}} s = t.$$

**Proposition 5.13** *Let $\overline{\Gamma}$ and $\overline{\Delta}$ be multisets of atomic ground formulas. Then the following two propositions are equivalent:*

1. *There is a proof of $\overline{\Gamma} \vdash \overline{\Delta}$ in $\overline{\mathfrak{G}}[\mathrm{E'G}]$.*
2. *$\overline{\Gamma} \cup \mathsf{AGrp} \models_{\mathrm{equ}} s = t$ for some $s = t \in \overline{\Delta}$.*

*where* $\mathsf{AGrp} \mathrel{\widehat{=}} \{(x+y)+z = x+(y+z),\ x+0 = x,\ x+(-x) = 0,\ x+y = y+x\}$.

Proof. We will implicitly use Theorem 5.12 several times in the proof. For the purpose of this proof, let $E \mathrel{\widehat{=}} \overline{\Gamma} \cup \mathsf{AGrp}$. We write $\overline{\Gamma} \overline{\vdash} \overline{\Delta}$ to mean that $\overline{\Gamma} \vdash \overline{\Delta}$ is provable in $\overline{\mathfrak{G}}[\mathrm{E'G}]$. We say that $\overline{\Gamma} \overline{\vdash} \overline{\Delta}$ holds by (ABS) if $\overline{\Gamma} \vdash \overline{\Delta}$ is an atomic basic sequent. We now consider each direction of the proposition in turn:

*2. implies 1.*: Assume $E \models_{\mathrm{equ}} s = t$ for some $s = t \in \overline{\Delta}$. We now proceed by structural induction over the proof of $E \vdash_{\mathrm{equ}} s = t$:

Case (QA): $E \vdash_{\mathrm{equ}} s = t$ because $(s = t) \in E$. Then $(s = t) \in \overline{\Gamma}$ (as $s = t$ is ground). Hence, we have $\overline{\Gamma} \overline{\vdash} \overline{\Delta}$ by (ABS).

Case (QR): $E \vdash_{\mathrm{equ}} s = t$ because $s$ is $t$. Then $\overline{\Gamma}, s = t \overline{\vdash} \overline{\Delta}$ by (ABS) and (E1) gives $\overline{\Gamma} \overline{\vdash} \overline{\Delta}$.

Case (QS): $E \vdash_{\mathrm{equ}} s = t$ because $E \vdash_{\mathrm{equ}} t = s$. By the induction hypothesis we get $\overline{\Gamma} \overline{\vdash} t = s, \overline{\Delta}$ and by (ABS) we have $\overline{\Gamma}, s = t \overline{\vdash} \overline{\Delta}$. Using (ES) we are done.

Case (QT): $E \vdash_{\mathrm{equ}} s = t$ because $E \vdash_{\mathrm{equ}} s = u$ and $E \vdash_{\mathrm{equ}} u = t$. By the induction hypothesis we get $\overline{\Gamma} \overline{\vdash} s = u, \overline{\Delta}$ and $\overline{\Gamma} \overline{\vdash} u = t, \overline{\Delta}$. Using (ABS) as in the previous case and (ET) we are done.

Case (QV): $E \vdash_{\mathrm{equ}} s = t$ because $s$ is $\sigma(s')$, $t$ is $\sigma(t')$ and $E \vdash_{\mathrm{equ}} s' = t'$ for some substitution $\sigma$ and formulas $s', t'$. As both $s$ and $t$ are ground we have that $s$ is $s'$ and $t$ is $t'$, hence we are done by induction.

Case (QC): $E \vdash_{\mathrm{equ}} s = t$ because $s$ is $f^n(s_1, \ldots, s_n)$, $t$ is $f^n(t_1, \ldots, t_n)$, and $E \vdash_{\mathrm{equ}} s_i = t_i$ for $1 \leq i \leq n$. By applying the induction hypothesis, (ABS) and (E2) we are done.

*1. implies 2.*: We proceed by structural induction over the proof of $\overline{\Gamma} \overline{\vdash} \overline{\Delta}$:

Case (ABS): $\overline{\Gamma} \vdash \overline{\Delta}$ because there is $s = t$ such that $s = t \in \overline{\Gamma}$ and $s = t \in \overline{\Delta}$. But then $s = t \in E$ and by (QA) we get $E \vdash_{\mathrm{equ}} s = t$.

Case (SD1): $\overline{\Gamma} \overline{\vdash} \overline{\Delta}$ because $((s+t)+r = s+(t+r)) \in \overline{\Delta}$ for some terms $s, t, r$. Define the substitution $\sigma \mathrel{\widehat{=}} [x \mapsto s, y \mapsto t, z \mapsto r]$. As $((x+y)+z = x+(y+z)) \in E$ we have (by (QA)) $E \vdash_{\mathrm{equ}} (x+y)+z = x+(y+z)$. Using (QV) we get $E \vdash_{\mathrm{equ}} \sigma((x+y)+z) = \sigma(x+(y+z))$, i.e. $E \vdash_{\mathrm{equ}} (s+t)+r = s+(t+r)$.

Case (SD2)–(SD4): Similar to (D1).

Case (E1): $\overline{\Gamma} \stackrel{=}{\vdash} \overline{\Delta}$ because $\overline{\Gamma}, t' = t' \stackrel{=}{\vdash} \overline{\Delta}$ for some term $t'$. By the induction hypothesis we have $E, t' = t' \models_{\text{equ}} s = t$ for some $s = t \in \overline{\Delta}$. From this we easily get $E \models_{\text{equ}} s = t$ since any structure satisfy $t' = t'$ (by definition).

Case (E2): $\overline{\Gamma} \stackrel{=}{\vdash} \overline{\Delta}$ because $\overline{\Gamma} \stackrel{=}{\vdash} s_i = t_i, \overline{\Delta}$ for terms $s_i$ and $t_i$ (where $1 \leq i \leq n$) and $\overline{\Gamma}, f^n(s_1, \ldots, s_n) = f^n(t_1, \ldots, t_n) \stackrel{=}{\vdash} \overline{\Delta}$ for some function symbol $f^n$. By the induction hypothesis we have $E \models_{\text{equ}} s = t$ for some $s = t \in \overline{\Delta}$, in which case we are done, or $E \models_{\text{equ}} s_i = t_i$ for all $i$, in which case we get $E \models_{\text{equ}} f^n(s_1, \ldots, s_n) = f^n(t_1, \ldots, t_n)$. Combining this with the last part of the induction hypothesis (viz., $E, f^n(s_1, \ldots, s_n) = f^n(t_1, \ldots, t_n) \models_{\text{equ}} s' = t'$ for some $s' = t' \in \overline{\Delta}$) we are done.

Case (EE): Similar to (E2). □

In the case of $\overline{\Delta}$ being singleton, *2.* above is a formulation of the decision problem known as *the word problem for finitely presented Abelian groups*. But this problem is known to be decidable [Ham88].

**Proposition 5.14**

- *The word problem for finitely presented groups is undecidable.*
- *The word problem for finitely presented* Abelian *groups is* decidable.

**Theorem 5.15** *Let* $\Gamma \vdash \Delta$ *be a sequent of* $\text{SIL}_{\text{QF}}$. *It is decidable whether there is a proof of* $\Gamma \vdash \Delta$ *in* $\mathfrak{G}$[LP′E′4′IAQG].

Proof. Perform a non-deterministic backwards proof search using only the rules of L4'IA. By the results of Section 5.1.1 this search will terminate. Now apply Lemma 5.11 and Proposition 5.13 and the theorem follows. □

As provability is undecidable in $\mathfrak{G}$[LP′E′4′IAQG, (Cut)] we thus have:

**Corollary 5.16** *Cut-elimination is not possible for* $\mathfrak{G}_{\text{SIL}}$.

## 5.3 Conclusion

If we revisit the discussion on the principles of the structure of sequent rules in Section 2.1.4, we see that both separation and explicitness fail for the SIL system. We almost achieve symmetry; only the monotonicity rules (LRM) break the symmetry.

The problem of not satisfying all these principles stems in the case of SIL from more fundamental difficulties with giving sequent calculus formulations to modal logics as we discussed in Section 3.4.1

The results of Section 5.2 tell us that any (quantifier-free) theorem in principle can be proved by splitting it in a number of lemmas, by means of (Cut), and then solve each of these lemmas automatically by the decision procedure sketched.

Cut-elimination is not possible for the SIL system. To cite Girard [Gir95]: "A sequent calculus without cut-elimination is like a car without engine". So, from a purely theoretical viewpoint the sequent calculus we have presented is not too interesting. But from a more pragmatic viewpoint, especially with respect to using a theorem prover for proving formulas of SIL, it still seems to be a more convenient system than the Hilbert system.

We have exclusively been concerned with SIL in this chapter. Due to the similarities of ITL and SIL, we could modify the sequent calculus and most of the results would (in slightly modified form) be applicable to ITL as well.

CHAPTER 6

# Labelled Natural Deduction

This chapter is concerned with developing and investigating a Labelled Natural Deduction (LND) system for SIL. This system turns out to be both "nicer" and "better" than the sequent calculus system of the previous section, both from a theoretical and a practical viewpoint.

We begin in Section 6.1 by developing a LND system for "pure" SIL. We consider normalization properties and prove that the LND system satisfies en extended subformula property. From a theoretical perspective this tells us that we have developed a "nice" system and from a more applied viewpoint, it has implications for proof search, as we will discuss.

In Section 6.2 we consider some extensions to the LND system. These include extending the duration domain to an infinite field as well as a LND DC extension.

Finally, in Section 6.3 we consider how the LND system for SIL can act as a general framework for other (similar) interval logics. We in particular discuss how ITL and NL can be "imitated" in the SIL system such that reasoning is done at a level which essentially hides the SIL specific details.

## 6.1 Labelled Signed Interval Logic

In this section we develop a LND system for SIL. First we consider extending the LND system for $\mathcal{L}^{\frown}_{\mathrm{AF}}$ to a first order version. Then we treat the full extension to SIL. Finally, we consider theoretical aspects of the system with regard to normalization, subformula property and proof search.

### 6.1.1 First Order Logic with Equality

In this section we extend the LND system for $\mathcal{L}^{\frown}_{\mathrm{AF}}$ to include first order logic with equality.

#### Rigidity and Chop-freeness

When we introduced the Hilbert system for SIL in Section 4.1.2 we stressed the necessity of additional sideconditions concerning rigidity and chop-freeness for some of the axioms. Noteworthy, these sideconditions were formulated at the meta level. The same principles were used in connection with the sequent calculus system (cf. Section 5.1).

Anticipating that we ultimately want to formalize the proof system in a theorem prover, we have to address the question of how to do this in the case of the rigidity and chop-freeness conditions. Both properties are syntactic by nature and can therefore easily be defined inductively over the structure of formulas (and terms). This is what we wish to do in this section in such a way that it fits nicely together with a LND system for SIL.

For this we introduce two new judgments, $\mathrm{ri}(\phi)$ and $\mathrm{cf}(\phi)$, stating, respectively, that $\phi$ is a rigid formula and that $\phi$ is a chop-free formula.

We start out be defining rules for chop-freeness. Below, $\oplus$ is placeholder for a binary Boolean operator, i.e., $\oplus \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$:

$$\frac{\mathrm{cf}(\phi) \quad \mathrm{cf}(\psi)}{\mathrm{cf}(\phi \oplus \psi)}\ \mathrm{cf}{\oplus}I \qquad \frac{\mathrm{cf}(\phi \oplus \psi)}{\mathrm{cf}(\phi)}\ \mathrm{cf}{\oplus}E \qquad \frac{\mathrm{cf}(\phi \oplus \psi)}{\mathrm{cf}(\psi)}\ \mathrm{cf}{\oplus}E$$

$$\frac{\mathrm{cf}(\phi)}{\mathrm{cf}(\neg\phi)}\ \mathrm{cf}{\neg}I \qquad \frac{\mathrm{cf}(\neg\phi)}{\mathrm{cf}(\phi)}\ \mathrm{cf}{\neg}E \qquad \frac{}{\mathrm{cf}(\phi)}\ \mathrm{cf}A$$

$$\frac{\mathrm{cf}(\phi)}{\mathrm{cf}((\forall \mathrm{x})\phi)}\ \mathrm{cf}{\forall}I \qquad \frac{\mathrm{cf}((\forall \mathrm{x})\phi)}{\mathrm{cf}(\phi)}\ \mathrm{cf}{\forall}E \qquad \frac{\mathrm{cf}(\phi)}{\mathrm{cf}((\exists \mathrm{x})\phi)}\ \mathrm{cf}{\exists}I \qquad \frac{\mathrm{cf}((\exists \mathrm{x})\phi)}{\mathrm{cf}(\phi)}\ \mathrm{cf}{\exists}E\ ,$$

with the following sidecondition:

- cf$A$: $\phi$ is atomic (e.g., $\bot$ or $s = t$).

We now turn to rules for rigidity. Below, $\otimes$ is a placeholder for a binary connective (i.e., we include modalities as well), that is, $\otimes \in \{\frown, \smile, \wedge, \vee, \rightarrow, \leftrightarrow\}$:

$$\frac{\mathrm{ri}(\phi) \quad \mathrm{ri}(\psi)}{\mathrm{ri}(\phi \otimes \psi)}\ \mathrm{ri}{\otimes}I \qquad \frac{\mathrm{ri}(\phi \otimes \psi)}{\mathrm{ri}(\phi)}\ \mathrm{ri}{\otimes}E \qquad \frac{\mathrm{ri}(\phi \otimes \psi)}{\mathrm{ri}(\psi)}\ \mathrm{ri}{\otimes}E$$

$$\frac{\mathrm{ri}(\phi)}{\mathrm{ri}(\neg\phi)}\ \mathrm{ri}\neg I \qquad \frac{\mathrm{ri}(\neg\phi)}{\mathrm{ri}(\phi)}\ \mathrm{ri}\neg E$$

$$\frac{\begin{array}{c}[\mathrm{ri(x)}]\\ \vdots\\ \mathrm{ri}(\phi)\end{array}}{\mathrm{ri}((\forall \mathrm{x})\phi)}\ \mathrm{ri}\forall I \qquad \frac{\begin{array}{c}[\mathrm{ri(x)}]\\ \vdots\\ \mathrm{ri}(\phi)\end{array}}{\mathrm{ri}((\exists \mathrm{x})\phi)}\ \mathrm{ri}\exists I$$

$$\frac{\mathrm{ri}((\forall \mathrm{x})\phi) \quad \mathrm{ri(s)}}{\mathrm{ri}(\phi[\mathrm{s/x}])}\ \mathrm{ri}\forall E \qquad \frac{\mathrm{ri}((\exists \mathrm{x})\phi) \quad \mathrm{ri(s)}}{\mathrm{ri}(\phi[\mathrm{s/x}])}\ \mathrm{ri}\exists E\ .$$

In the case of the definition of chop-freeness we only had to consider the formula level as — without exception — any atomic formula is chop-free. The case for rigidity is a little more complicated; here we have to consider the term-level as well. We overload the ri judgment for this. If $\odot$ is a rigid binary predicate/function symbol (e.g., $=$ or $+$), and $\star$ is a rigid unary predicate/function symbol, we have the following rules:[1]

$$\frac{\mathrm{ri(s)} \quad \mathrm{ri(t)}}{\mathrm{ri}(\mathrm{s}\odot\mathrm{t})}\ \mathrm{ri}\odot I \qquad \frac{\mathrm{ri}(\mathrm{s}\odot\mathrm{t})}{\mathrm{ri(s)}}\ \mathrm{ri}\odot E \qquad \frac{\mathrm{ri}(\mathrm{s}\odot\mathrm{t})}{\mathrm{ri(t)}}\ \mathrm{ri}\odot E$$

$$\frac{\mathrm{ri(s)}}{\mathrm{ri}(\star\mathrm{s})}\ \mathrm{ri}\star I \qquad \frac{\mathrm{ri}(\star\mathrm{s})}{\mathrm{ri(s)}}\ \mathrm{ri}\star E\ .$$

Besides these rules we must state explicitly when certain atomic formulas and constants are rigid, e.g., that $\mathrm{ri}(\bot)$ and $\mathrm{ri}(0)$.

In this section, so far, all rules have been exclusively concerned with the judgments cf and ri. We now present an important rule relating a ri judgment and a labelled formula to a labelled formula:

$$\frac{v:\phi \quad \mathrm{ri}(\phi)}{w:\phi}\ R\ .$$

This rule expresses the semantic notion of rigid formulas being interpreted the same way in all (possible) worlds.

### Quantifiers and Equality

We now consider LND rules for the quantifiers. These rules are essentially the ones given in Section 2.2.5 augmented with labels in a straightforward way. Furthermore,

[1] It would be straightforward to define rules for predicate/function symbols of arity more than two if needed.

the sideconditions concerning chop-freeness and rigidity are made explicit and concrete parts of the rules, as anticipated in the previous section. The structure is recognizable when comparing with the quantifier rules for the Hilbert and sequent calculus systems:

$$\frac{\begin{array}{c}[\mathrm{ri(x)}]\\ \vdots\\ w:\phi\end{array}}{w:(\forall x)\phi}\ \forall I \qquad \frac{w:(\forall x)\phi \quad \mathrm{ri(s)}}{w:\phi[s/x]}\ \forall E_{\mathrm{ri}} \qquad \frac{w:(\forall x)\phi \quad \mathrm{cf}(\phi)}{w:\phi[s/x]}\ \forall E_{\mathrm{cf}}$$

$$\frac{w:\phi[s/x] \quad \mathrm{ri(s)}}{w:(\exists x)\phi}\ \exists I_{\mathrm{ri}} \qquad \frac{w:\phi[s/x] \quad \mathrm{cf}(\phi)}{w:(\exists x)\phi}\ \exists I_{\mathrm{cf}} \qquad \frac{w:(\exists x)\phi \qquad \begin{array}{c}[w:\phi][\mathrm{ri(x)}]\\ \vdots\\ v:\psi\end{array}}{v:\psi}\ \exists E\ ,$$

with the following sideconditions:

- $\forall I$: $x$ is not free in any assumption on which $\phi$ depends except ri(x).
- $\exists E$: $x$ is not free in $\psi$ nor in any assumption on which $\psi$ depends except $\phi$ and ri(x).

As is well-known from classical logic, the rules for $\exists$ are derivable from those for $\forall$ (and vice versa) — this still holds for the above modified rules. Thus, we can restrict attention to $\forall$ in the following if we wish so. A contraction step for $\forall$ (cf. Section 3.5.1) can be defined and Theorem 3.14 can be modified accordingly. Definition 3.15 can be extended to include the case for $\forall$ too. (See, e.g., [TS96].)

The form of these quantifier rules is related to the notion of *free logics* [Ben86] where a term used for instantiation must exist and be explicitly stated in the rules.

We now consider equality rules. These are again as in Section 2.2.5 but augmented in a way similar to the quantifier rules:

$$\frac{w:\phi[s/x] \quad w:s=t \quad \mathrm{ri(s)} \quad \mathrm{ri(t)}}{w:\phi[t/x]}\ \mathrm{Subst}_{\mathrm{ri}} \qquad \frac{}{w:s=s}\ \mathrm{Refl}$$

$$\frac{w:\phi[s/x] \quad w:s=t \quad \mathrm{cf}(\phi)}{w:\phi[t/x]}\ \mathrm{Subst}_{\mathrm{cf}}\ .$$

**Lemma 6.1** *The rules* $\mathrm{Subst}_{\mathrm{ri}}$ *and* $\mathrm{Subst}_{\mathrm{cf}}$ *can can be restricted to applications where* $\phi$ *is atomic.*

Proof. The basic idea is the same as that of the proof of 1) in Proposition 3.12.

First we consider a derivation containing applications of $\mathrm{Subst}_{\mathrm{ri}}$: Pick out an application where the conclusion has maximal size. If not atomic (in which case

we are done), this conclusion will have form $\phi \to \psi$, $(\forall x)\phi$ or $\phi^\frown\psi$. Below we only consider the latter case (the first two cases follow analogously). We replace the derivation with one where the affected $\text{Subst}_{\text{ri}}$ is replaced by two applications of $\text{Subst}_{\text{ri}}$ where the conclusions have less size:

$$\dfrac{w : (\phi^\frown\psi)[s/x] \quad w : s = t \quad \text{ri(s)} \quad \text{ri(t)}}{w : (\phi^\frown\psi)[t/x]}\ \text{Subst}_{\text{ri}} \qquad \rightsquigarrow$$

$$\dfrac{\dfrac{w : (\phi^\frown\psi)[s/x] \quad \Pi \quad [\mathcal{R}(v,u,w)]^1}{u : \psi[s/x]} \quad \dfrac{\qquad\qquad w : s = t \quad \text{ri(s)} \quad \text{ri(t)}}{u : \psi[t/x]}\ \text{Subst}_{\text{ri}}}{w : (\phi^\frown\psi)[t/x]}\ {}^\frown I^1 \quad ,$$

where $(\phi^\frown\psi)[s/x] \equiv (\phi[s/x])^\frown(\psi[s/x])$ and $\Pi$ is

$$\dfrac{[v : \phi[t/x]]^1 \quad \dfrac{w : s = t}{w : t = s} \quad \text{ri(s)} \quad \text{ri(t)}}{v : \phi[s/x]}\ \text{Subst}_{\text{ri}} \quad .$$

By induction it is now easy to see that repeated applications of this transformation yield the desired derivation.

The principles of the proof are the same in the case of $\text{Subst}_{\text{cf}}$ but we do here not have to consider the case where the maximal formula has form $\phi^\frown\psi$ because of the sidecondition concerning chop-freeness. □

A corollary of this lemma is that we could actually do without the $\text{Subst}_{\text{ri}}$ rule completely. An atomic formula is always chop-free and $\text{Subst}_{\text{cf}}$ therefore suffices.

### Structure of Derivations Involving cf/ri Judgments

By inspecting the rules involving both labelled formulas and cf/ri judgments we observe that the derivation of a labelled formula might depend on the ri/cf rules whereas derivations of ri/cf judgments never depend on labelled formulas or each other; thus, the full derivation tree of a labelled formula can be seen as "decorated" with independent derivations of ri/cf judgments.

Having this way isolated the derivations involving only ri/cf judgments we can now consider the structure of these "pure" ri/cf derivations. The notions of maximal formula, contraction step and normal derivation can easily be carried over to this setting. Because the ri/cf rules have very simple and similar forms it should thus be fairly easy to realize that any ri/cf derivation can be normalized. Furthermore, such normal derivations will satisfy a subformula property. This entails that the question of whether a given ri/cf judgment is derivable from a set of ri/cf judgments is decidable. This is naturally an important and useful result for proof search in the full LND system.

### 6.1.2 Signed Interval Logic

The previous section provided us with LND rules for a first order extension including the judgments ri/cf. We adjoin this extension to the LND system for $\mathcal{L}_{\mathrm{AF}}^{\frown}$ introduced in Section 3.5.1 to get a first order $\mathcal{L}_{\mathrm{AF}}^{\frown}$ LND system.

Our goal in this section is to further extend this system to a full LND system for SIL. The first important aspect to address is the structure of the worlds which make up the labels of the logic: We are now working with square models which means that the labels are pairs related as defined in Definition 3.4, i.e., $((i,k),(k,j),(i,j)) \in R$. This in turn means that we do not have to include the $\mathcal{R}$ judgments explicitly in the system but can express the relations among worlds implicitly.

In the case of the rules $\frown I$, $\frown E$, $\smile I$ and $\smile E$ we get (cf. the original definition of the rules in Section 3.5.1):

$$\frac{(i,k):\phi \quad (k,j):\psi}{(i,j):\phi\frown\psi}\ \frown I \qquad \frac{(i,j):\phi\frown\psi \qquad \begin{array}{c}[(i,k):\phi]\ [(k,j):\psi]\\ \vdots \\ (m,n):\chi\end{array}}{(m,n):\chi}\ \frown E$$

$$\frac{\begin{array}{c}[(i,k):\phi]\\ \vdots \\ (k,j):\psi\end{array}}{(i,j):\phi\smile\psi}\ \smile I \qquad \frac{(i,j):\phi\smile\psi \quad (i,k):\phi}{(k,j):\psi}\ \smile E\ ,$$

with the following sideconditions:

- $\frown E$: $k$ is different from $i,j,n,m$, and does not occur in any assumption on which the upper occurrence of $(m,n):\chi$ depends except $(i,k):\phi$ and $(k,j):\psi$.
- $\smile I$: $k$ is different from $i$ and $j$, and does not occur in any assumption on which $(k,j):\psi$ depends except $(i,k):\phi$.

We now define the additional LND rules necessary for the full SIL system. First we include two important uniqueness rules:

$$\frac{(l,j):\phi \quad (i,k):\ell=s \quad (i,l):\ell=s \quad \mathrm{ri(s)}}{(k,j):\phi}\ S1$$

$$\frac{(i,l):\phi \quad (k,j):\ell=s \quad (l,j):\ell=s \quad \mathrm{ri(s)}}{(i,k):\phi}\ S2\ .$$

These rules are related to a definition which can be found in the detailed completeness proof for SIL in [Ras99b]. The rules can be seen as a way of expressing when two intervals are essentially the same, see Figure 6.1.

Furthermore, we need rules defining the basic properties of $\ell$:

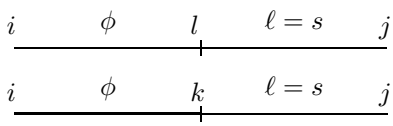



Figure 6.1: The rule $S2$ states that if $\phi$ holds on $(i,l)$ and $\ell = s$ holds on both $(l,j)$ and $(k,j)$ then $\phi$ holds on $(i,k)$ as well. Similarly for $S1$.

$$\frac{(i,k):\ell=s \quad (k,j):\ell=t \quad \text{ri(s)} \quad \text{ri(t)}}{(i,j):\ell=s+t}\ \ell{+}I$$

$$\frac{}{(i,i):\ell=0}\ \ell 0 \qquad \frac{\begin{array}{c}[(i,k):\ell=s]\\ [(k,j):\ell=t]\\ \vdots\\ (i,j):\ell=s+t \quad \text{ri(s)} \quad \text{ri(t)} \quad (m,n):\chi\end{array}}{(m,n):\chi}\ \ell{+}E\ ,$$

with the following sideconditions:

- $\ell{+}E$: $k$ is different from $i,j,m,n$, and does not occur in any assumption on which the upper occurrence of $(m,n):\chi$ depends, except $(i,k):\phi$ and $(k,j):\psi$.

Finally, we have axioms expressing the properties of an Abelian group, viz.

$$\frac{}{(i,j):s+(t+u)=(s+t)+u}\ \text{LD1} \qquad \frac{}{(i,j):s+0=s}\ \text{LD2}$$

$$\frac{}{(i,j):s+-s=0}\ \text{LD3} \qquad \frac{}{(i,j):s+t=t+s}\ \text{LD4}\ .$$

The semantics of SIL can straightforwardly be modified to a labelled semantics in the same way as a labelled semantics was given for $\mathcal{L}_{\widehat{\text{AF}}}$ (cf. Section 3.5.1).

**Proposition 6.2**

$\models_{\text{SIL}} \phi$ *iff* $\Vdash_{\text{SIL}} (i,j):\phi$ *for all* $i,j$.

We now come to the crucial theorem of this section.

**Theorem 6.3**

$\models_{\text{SIL}} \phi$ *iff* $\vdash^{\text{LND}}_{\text{SIL}} (i,j):\phi$ *for all* $i,j$.

Proof. The proof extends the proof of Proposition 3.11. As there, soundness is straightforward. For completeness we need to show that the additional axioms of the

SIL system (cf. Section 4.1.2) are derivable in the LND system. This is not difficult. As an example we derive the axiom L1:

$$\dfrac{\dfrac{\dfrac{[(i,j):(\ell=s)^\frown(\phi\to\bot)]^3 \quad \dfrac{\Pi \quad [(m,j):\phi\to\bot]^4}{(m,j):\bot}\to E}{(i,j):\bot}\perp E}{(i,j):(\ell=s)^\frown(\phi\to\bot)\to\bot}{}^\frown E^4 \quad [(i,j):(\ell=s)^\frown\phi]^1}{(i,j):(\ell=s)^\frown(\phi\to\bot)\to\bot}{}^\frown E^2}{(i,j):(\ell=s)^\frown\phi\to((\ell=s)^\frown(\phi\to\bot)\to\bot)}\to I^1 \quad \to I^3 \quad ,$$

where $\Pi$ is

$$\dfrac{[(i,m):\ell=s]^4 \quad [(i,k):\ell=s]^2 \quad [(k,j):\phi]^2 \quad \text{ri(s)}}{(m,j):\phi}\,S1 \quad .$$

Note how the rule $S1$ is utilized in the derivation. □

**Lemma 6.4** *The rules $R$, $S1$ and $S2$ can be restricted to applications where $\phi$ is atomic.*

Proof. This lemma is proved in basically the same way as Lemma 6.1. However, it is worth noticing that the lemma does not hold for the general version of $R$ (with general accessibility relations) but when we work in square frames it does. □

Finally, we can show the following result:

**Lemma 6.5** *The rule $\ell{+}E$ can be restricted to applications where $\chi$ is $\bot$.*

Proof. For the proof of this lemma we assume that we are considering the $^\smile$-fragment of the LND system. We then show that in the only case where $\ell{+}E$ is utilized in the completeness proof (Theorem 6.3), $\chi$ can be taken to be $\bot$. The $\ell{+}E$ rule is used for proving the left-to-right direction of L2 as follows (note the alternative form of L2 due to the fact that $^\smile$ is used):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[(i,j):(\ell=s)^\smile((\ell=t)\to\bot)]^2 \quad [(i,k):\ell=s]^3}{(k,j):(\ell=t)\to\bot}\smile E \quad [(k,j):\ell=t]^3}{(k,j):\bot}\to E}{(i,j):\bot}\bot E \quad [(i,j):\ell=s+t]^1}{(i,j):\bot}\ell{+}E^3}{(i,j):(\ell=s)^\smile((\ell=t)\to\bot)\to\bot}\to I^2}{(i,j):\ell=s+t\to((\ell=s)^\smile((\ell=y)\to\bot)\to\bot)}\to I^1 \quad .$$

For readability we left out the judgments concerning ri in the above proof. □

Lemmas 6.1, 6.4 and 6.5 are utilized in the normalization result below.

### 6.1.3 Normalization

In this section we consider normalization properties of the full LND system for SIL.

For this, we extend the definition of a normal derivation (Definition 3.13) to include the requirements that for applications of the rules $\mathrm{Subst}_{\mathrm{ri}}$, $\mathrm{Subst}_{\mathrm{cf}}$, $R$, $S1$ and $S2$, $\phi$ has to be atomic, and that for applications of $\ell{+}E$, $\chi$ has to be $\bot$. By the Lemmas 6.1, 6.4 and 6.5 we have that Theorem 3.14 is valid for the full SIL system too.

Unfortunately, we will not have as nice properties of tracks as those of Proposition 3.16. But because of the structure of normal derivations, tracks can still be divided into three main parts: An elimination part, a part working on atomic formulas and/or $\bot$, and an introduction part.

We will below go into a little more detail concerning the structure of the middle part, but the important thing is that a normal derivation as defined is enough to achieve an interesting version of the subformula property.

First, though, we have to address what we mean by subformula in a first order logic with equality: We say that $\phi[s/x]$ is a subformula of $\psi[t/x]$ if $\phi$ is a subformula of $\psi$, independently of $t$ and $s$. In other words, we do not take the term level into account.

We extend the definition of the subformula property of a derivation (Definition 3.17) to include the case where a formula in a derivation is allowed to be an arbitrary atomic formula.

**Definition 6.6** Consider a derivation $\Pi$ of $w:\alpha$ from $\Gamma$ and $\Delta$. Let $S=\{\alpha\}\cup\{\gamma \mid u:\gamma\in\Gamma \text{ for some } u\}$. Then $\Pi$ is said to have the *extended subformula property* if for any labelled formula $v:\beta$ of $\Pi$, $\beta$ is

1. an atomic formula,
2. $\bot$,

3. a subformula of some formula in $S$, or

4. $\neg\beta'$ and $\beta'$ is a subformula of some formula in $S$.

We then have the following main result.

**Theorem 6.7** *Any normal derivation in the LND system for SIL satisfies the extended subformula property.*

Note that this result relies on the fact that with the above definition of the subformula property it is not a problem to add axioms to the system as long as they are atomic. This is in particular the case for the SIL system.

Let us now look a little more closely on the middle part of a track in a normal derivation: An arbitrary (finite) sequence of occurrences of the rules $R$, $\mathrm{Subst}_{\mathrm{cf}}$,[2] $S1$ and $S2$ can be transformed to a sequence of the following form: First, one occurrence of $\mathrm{Subst}_{\mathrm{cf}}$, then one of $R$, and finally a sequence of occurrences of $S1$ followed by a sequence of $S2$'s. In other words, all four rules can switch places in the sequence, and $\mathrm{Subst}_{\mathrm{cf}}$ and $R$ can furthermore be contracted to just one application.

It is clear that we can assume the conclusions of $R$, $\mathrm{Subst}_{\mathrm{cf}}$, $S1$ and $S2$ *not* to be $\bot$: $\mathrm{Subst}_{\mathrm{cf}}$ will not have any affect (thus it can be removed completely) and $R$, $S1$, $S2$ can be replaced by $\bot E$. As a consequence, in the middle part of a track, an occurrence of $\bot E$ will potentially appear as the first rule and follow each occurrence of $\ell{+}E$ except the last occurrence of $\ell{+}E$.

The definition of maximality could be extended to include a $\ell{+}I$ rule followed by a $\ell{+}E$ rule — which are easily seen to be contractable.

We could continue (for a while, at least) making such observations concerning the structure of the middle part of a track but as we will not later utilize these results, we choose instead to consider the main normalization result a little closer.

### Proof Search

What are the main consequences of the normalization result, in particular Theorem 6.7? Firstly, it convincingly indicates that we, from a theoretical perspective, have defined a "proper" proof system for SIL. Secondly, it has direct implications for proof construction/search: Theorem 6.7 tells us that we can prove any theorem $\phi$ of SIL by only considering subformulas of $\phi$.

A good way of illustrating the latter point is to consider a *labelled* sequent calculus based on the LND system for SIL. For this, we include labelled versions of the sequent rules for first order logic with equality considered in Section 5.1. But instead of all the rules there involving chop, we now only have the following two rules:

[2] Because of the corollary of Lemma 6.5 we do not consider $\mathrm{Subst}_{\mathrm{ri}}$.

$$\frac{\Gamma,\ (i,k):\phi,\ (k,j):\psi\ \vdash\ \Delta}{\Gamma,\ (i,j):\phi^\frown\psi\ \vdash\ \Delta}\ (\mathrm{L}^\frown)$$

$$\frac{\Gamma\ \vdash\ (i,k):\phi,\ \Delta\quad \Gamma\ \vdash\ (k,j):\psi,\ \Delta}{\Gamma\ \vdash\ (i,j):\phi^\frown\psi,\ \Delta}\ (\mathrm{R}^\frown)\ .$$

Hence, we have "proper" R-/L-rules for chop.

For a complete labelled sequent calculus we also need sequent versions of the LND rules $R$, $S1$, $S2$, $\ell + I$ and $\ell + E$ as well as the axioms.

We see that both $(\mathrm{L}^\frown)$ and $(\mathrm{R}^\frown)$ satisfy the standard subformula property. Thus, a backwards proof search using the propositional rules together with $(\mathrm{L}^\frown)$ and $(\mathrm{R}^\frown)$ will terminate and we are left with term-level reasoning over equalities. This gives an operational explanation of the normalization result above.

In Chapter 9 we will see how these principles show for actual proof making in the LND system encoded in a theorem prover (Isabelle).

Finally, to formally prove that the sketched labelled sequent calculus system in fact is equivalent to the LND system, it would be most convenient to include the (Cut) rule in the sequent system. But we claim that (Cut) can be eliminated afterwards due to the well-known correspondence between cut-elimination and normalization [TS96].

## 6.2 Extensions to Labelled SIL

In the previous section we saw how a sound and complete LND system for SIL could be defined. In this section we consider some extensions to that system.

### 6.2.1 Axioms for Order and Arithmetic

It is not difficult to extend the LND system to a totally ordered version. Simply add the following labelled axioms and rules (corresponding to the Hilbert axioms D5–D9 listed in Section 4.1.4):

$$\frac{}{(i,j):s\le s}\ \mathrm{LD5}\qquad \frac{(i,j):s\le t\quad (i,j):t\le u}{(i,j):s\le u}\ \mathrm{LD6}$$

$$\frac{(i,j):s\le t\quad (i,j):t\le s}{(i,j):s=t}\ \mathrm{LD7}\qquad \frac{}{(i,j):s\le t\vee t\le s}\ \mathrm{LD8}$$

$$\frac{(i,j):s\le t}{(i,j):s+u\le t+u}\ \mathrm{LD9}\ .$$

It is now fairly easy to show the following result (cf. Theorem 6.3).

**Theorem 6.8**

$$\models_{\mathrm{SIL_{to}}} \phi \quad \textit{iff} \quad \vdash^{\mathrm{LND}}_{\mathrm{SIL_{to}}} (i,j) : \phi \quad \textit{for all } i,j.$$

For actual use of the LND system for specification and verification purposes, it would be wishful to be able to reason in full arithmetic — and not just a totally ordered Abelian group. But it is well-known that it is impossible to axiomatize aritmetic; what we do here is to add some sound properties which are often useful.

First, we wish to specify that our domain is infinite (both "upwards" and "downwards" with respect to $<$). This can be done by means of the so-called Archimedian axioms [Fuc63]:

$$\frac{}{(i,j) : (\exists x) x < s} \qquad \frac{}{(i,j) : (\exists x) s < x} \ .$$

Second, we would like to introduce multiplication $\circ$ by extending the Abelian group to a field:

$$\frac{}{(i,j) : s \circ t = t \circ s} \qquad \frac{}{(i,j) : s \circ 1 = s} \qquad \frac{(i,j) : s \neq 0}{(i,j) : s \circ \frac{1}{s} = 1}$$

$$\frac{}{(i,j) : (s \circ t) \circ u = s \circ (t \circ u)} \qquad \frac{}{(i,j) : s \circ (t + u) = s \circ t + s \circ u} \ .$$

Finally, if we add congruence rules relating multiplication and reciprocial to the order relation $\leq$, we have collectively axiomatized a *totally ordered infinite field*:

$$\frac{(i,j) : 0 < u \quad (i,j) : s \leq t}{(i,j) : s \circ u \leq t \circ u} \qquad \frac{(i,j) : 0 < s}{(i,j) : 0 < \frac{1}{s}} \ .$$

The use, relevance and consequences of these extensions are further discussed in Chapters 8 and 9 in connection with the Isabelle encoding.

### 6.2.2 Labelled SDC

A labelled SDC extension to labelled SIL is not difficult to define.

All SDC axioms listed in Section 4.3.1 are simply added straightforwardly as labelled axioms. The only actual modification is in the case of DA5 which is converted into a rule in a form similar to $\ell{+}I$ (this makes sure that $\frown$ still only occurs in $\frown I$ and $\frown E$):

$$\frac{(i,k) : \textstyle\int S = s \quad (k,j) : \int S = t \quad \mathrm{ri(s)} \quad \mathrm{ri(t)}}{(i,j) : \int S = s + t}\ \mathrm{LDA5} \ .$$

The induction rules are added as labelled versions without further modifications as well. Equivalence of the systems is trivial to prove.

## 6.3 Labelled SIL as a General Framework

We have so far in this chapter concentrated on developing and investigating a LND system for SIL. In this section we will consider how LND systems for ITL and NL can be defined.

First we briefly sketch how systems similar to that for SIL could be defined for ITL and NL directly. Then we more detailed discuss how the LND system for SIL can act as a framework in which ITL and NL systems can be imitated.

### 6.3.1 LND systems for ITL and NL

We saw in Section 4.2.1 how the semantics of $\frown$ for ITL can be regarded as a restricting of that of $\frown$ for SIL: The $k$ in the semantic definition for $\frown$ has to fulfill $i \leq k \leq j$ and is thus not completely arbitrary.

This can be reflected in a LND system for ITL if we introduce a judgment $\sqsubseteq$ expressing the ordering on the temporal domain. Thus, we define the following rules (where $i \sqsubseteq j$ is assumed):

$$\frac{(i,k):\phi \quad (k,j):\psi \quad i \sqsubseteq k \quad k \sqsubseteq j}{(i,j):\phi\frown\psi}\ \frown I \qquad \frac{(i,j):\phi\frown\psi \qquad \begin{array}{c}[(k,j):\psi]\ [k \sqsubseteq j] \\ [(i,k):\phi]\ [i \sqsubseteq k] \\ \vdots \\ (m,n):\chi\end{array}}{(m,n):\chi}\ \frown E\ .$$

Similar restrictions to other rules of SIL would give us a LND system for ITL if we furthermore add rules defining the properties of the $\sqsubseteq$ judgment (such as reflexivity and transitivity).

The same exercise could be carried out for NL. The structure of the rules would be indicated by the semantic definitions of the two unary modalities of NL, cf. Section 4.2.2.

We will not go further into such considerations here as a much easier way of having LND systems for ITL and NL will be discussed in the following section.

### 6.3.2 A General Framework

In this section we consider how to "imitate" a LND system for ITL, subsequently NL, within the LND system for $\mathrm{SIL}_{\mathrm{to}}$.

One of the reasons making this possible, is the fact that it is actually not necessary to introduce the $\sqsubseteq$ judgment explicitly: Definition 4.17 tells us that $i \sqsubseteq j$ is

semantically equivalent to $(i,j) : \mathit{fwd}$, and we can therefore use the latter judgment instead.

In Section 4.2.1 we mentioned how we could define the chop of ITL by means of the chop of SIL, within SIL: $\phi \mathbin{\overline{\frown}} \psi \;\widehat{=}\; (\phi \wedge \mathit{fwd}) \frown (\psi \wedge \mathit{fwd})$. We can now formulate the connection formally in the following important theorem (where $\vdash_{\mathrm{ITL}}$ denotes theoremhood in the Hilbert system for ITL):

**Theorem 6.9**

$$\vdash_{\mathrm{ITL}} \phi \quad \textit{iff} \quad (i,j) : \mathit{fwd} \vdash^{\mathrm{LND}}_{\mathrm{SIL_{to}}} (i,j) : \overline{\phi} \quad \textit{for all } i,j,$$

*where $\overline{\phi}$ is $\phi$ with all occurrences of $\frown$ replaced by $\mathbin{\overline{\frown}}$.*

Proof. For the *if*-direction we use a semantic argument: First, we observe that the righthand-side is equivalent to $\vdash^{\mathrm{LND}}_{\mathrm{SIL_{to}}} (i,j) : \mathit{fwd} \rightarrow \overline{\phi}$. By Theorem 6.8 this is equivalent to $\models_{\mathrm{SIL_{to}}} \mathit{fwd} \rightarrow \overline{\phi}$. Hence, the soundness and completeness result for ITL [Dut95a] entails that we are done if we can establish the following:

$$\models_{\mathrm{ITL}} \phi \quad \text{if} \quad \models_{\mathrm{SIL_{to}}} \mathit{fwd} \rightarrow \overline{\phi}. \tag{6.1}$$

For this we assume that $\phi$ is not valid in ITL and show that this implies that $\mathit{fwd} \rightarrow \overline{\phi}$ is not valid in $\mathrm{SIL_{to}}$ either. Our assumption implies that there is some ITL model $\mathfrak{M}$, valuation $\mathcal{V}$ and interval $(i,j)$ of $\mathfrak{M}$ (with $i \leq j$) which does not satisfy $\phi$ (written $\mathfrak{M}, \mathcal{V}, (i,j) \not\models_{\mathrm{ITL}} \phi$). Based on this, we now wish to construct a $\mathrm{SIL_{to}}$ model $\mathfrak{M}'$, valuation $\mathcal{V}'$ and interval $(i',j')$ of $\mathfrak{M}'$ and show that

$$\mathfrak{M}', \mathcal{V}', (i',j') \not\models_{\mathrm{SIL_{to}}} \mathit{fwd} \rightarrow \overline{\phi},$$

which implies (6.1). To construct $\mathfrak{M}'$, we in a straightforward way extend $\mathfrak{M}$ to allow for backward intervals. The interpretation of function/predicate symbols of $\mathfrak{M}'$ is the same as that of $\mathfrak{M}$ on forward intervals and arbitrary on backward intervals. The valuation $\mathcal{V}'$ and interval $(i',j')$ are taken to be the same as $\mathcal{V}$ and $(i,j)$, respectively. As $i \leq j$ we have $\mathfrak{M}', \mathcal{V}, (i,j) \models_{\mathrm{SIL_{to}}} \mathit{fwd}$, thus $\mathfrak{M}', \mathcal{V}, (i,j) \not\models_{\mathrm{SIL_{to}}} \mathit{fwd} \rightarrow \overline{\phi}$ is equivalent to $\mathfrak{M}', \mathcal{V}, (i,j) \not\models_{\mathrm{SIL_{to}}} \overline{\phi}$.

We now show that

$$\mathfrak{M}, \mathcal{V}, (i,j) \not\models_{\mathrm{ITL}} \phi \quad \text{iff} \quad \mathfrak{M}', \mathcal{V}, (i,j) \not\models_{\mathrm{SIL_{to}}} \overline{\phi},$$

by structural induction over $\phi$. The base case is straightforward. The only interesting case of the induction step is the one where $\phi$ is $\phi_1 \frown \phi_2$.

First the left-to-right direction. That

$$\mathfrak{M}, \mathcal{V}, (i,j) \not\models_{\mathrm{ITL}} \phi_1 \frown \phi_2,$$

means that for all $k$, $i \leq k \leq j$, $\mathfrak{M}, \mathcal{V}, (i,k) \not\models_{\mathrm{ITL}} \phi_1$ or $\mathfrak{M}, \mathcal{V}, (k,j) \not\models_{\mathrm{ITL}} \phi_2$. By induction this means $\mathfrak{M}', \mathcal{V}, (i,k) \not\models_{\mathrm{SIL_{to}}} \overline{\phi_1}$ or $\mathfrak{M}', \mathcal{V}, (k,j) \not\models_{\mathrm{SIL_{to}}} \overline{\phi_2}$, which again

means $\mathfrak{M}', \mathcal{V}, (i,k) \not\models_{\mathrm{SIL_{to}}} \overline{\phi_1} \wedge \mathit{fwd}$ or $\mathfrak{M}', \mathcal{V}, (k,j) \not\models_{\mathrm{SIL_{to}}} \overline{\phi_2} \wedge \mathit{fwd}$. For $k < i$ or $j < k$ the latter is the case as well. This means that

$$\mathfrak{M}', \mathcal{V}, (i,j) \not\models_{\mathrm{SIL_{to}}} (\overline{\phi_1} \wedge \mathit{fwd}) \frown (\overline{\phi_1} \wedge \mathit{fwd}),$$

which finally gives us $\mathfrak{M}', \mathcal{V}, (i,j) \not\models_{\mathrm{SIL_{to}}} \overline{\phi}$.

Now the right-to-left direction. That

$$\mathfrak{M}', \mathcal{V}, (i,j) \not\models_{\mathrm{SIL_{to}}} \overline{\phi_1} \mathbin{|\!\frown\!|} \overline{\phi_2},$$

means that for all $k$, $\mathfrak{M}', \mathcal{V}, (i,k) \not\models_{\mathrm{SIL_{to}}} \overline{\phi_1} \wedge \mathit{fwd}$ or $\mathfrak{M}', \mathcal{V}, (k,j) \not\models_{\mathrm{SIL_{to}}} \overline{\phi_2} \wedge \mathit{fwd}$. For $i \leq k \leq j$ this means that $\mathfrak{M}', \mathcal{V}, (i,k) \not\models_{\mathrm{SIL_{to}}} \overline{\phi_1}$ or $\mathfrak{M}', \mathcal{V}, (k,j) \not\models_{\mathrm{SIL_{to}}} \overline{\phi_2}$, which by induction entails $\mathfrak{M}, \mathcal{V}, (i,k) \not\models_{\mathrm{ITL}} \phi_1$ or $\mathfrak{M}, \mathcal{V}, (k,j) \not\models_{\mathrm{ITL}} \phi_2$. Finally, we have that $\mathfrak{M}, \mathcal{V}, (i,j) \not\models_{\mathrm{ITL}} \phi$.

This concludes the *if*-direction of the theorem. We now consider the *only if*-direction. In Section 4.2.1 we mentioned how the Hilbert system for SIL could be converted into that for ITL by just adding the axiom $\ell \geq 0$ and slightly modifying the L2 axiom. Given this, we proceed by induction over the length of the ITL Hilbert proof. This amounts to showing that all ITL axioms and Hilbert rules are provable (only assuming *fwd*) in the LND system for SIL where, importantly, all occurrences of $\frown$ have been replaced by $\mathbin{|\!\frown\!|}$. This is not difficult. □

As $\mathbin{|\!\frown\!|}$ is nothing but an abbreviation it would of course be somewhat cumbersome to have to expand it to reason within the $\mathrm{SIL_{to}}$ system. Furthermore, the connection to the explicitly stated rules for ITL in the previous section would then not be clear. Fortunately, we can easily derive "proper" I- and E-rules for $\mathbin{|\!\frown\!|}$ within the SIL system as follows:

$$\frac{(i,k) : \phi \quad (k,j) : \psi \quad (i,k) : \mathit{fwd} \quad (k,j) : \mathit{fwd}}{(i,j) : \phi \mathbin{|\!\frown\!|} \psi} \mathbin{|\!\frown\!|} I$$

$$\frac{\begin{array}{c}(i,j) : \phi \mathbin{|\!\frown\!|} \psi \quad (i,j) : \mathit{fwd} \\ \\ \\ \end{array} \quad \begin{array}{c} \\ [(k,j) : \psi] \; [(k,j) : \mathit{fwd}] \\ [(i,k) : \phi] \; [(i,k) : \mathit{fwd}] \\ \vdots \\ (m,n) : \chi \end{array}}{(m,n) : \chi} \mathbin{|\!\frown\!|} E \quad .$$

Note the great similarity with the rules of the previous section. In practice, one would probably not feel any difference at all when using the rules.

We now state a result for NL similar to that for ITL. (Cf. Section 4.2.2.) The proof follows the exact same lines as that of Theorem 6.9.

**Theorem 6.10**

$$\vdash_{\mathrm{NL}} \phi \quad \textit{iff} \quad (i,j) : \mathit{fwd} \vdash^{\mathrm{LND}}_{\mathrm{SIL_{to}}} (i,j) : \overline{\phi} \quad \textit{for all } i,j,$$

*where $\overline{\phi}$ is $\phi$ with all occurrences of $\Diamond_l$ and $\Diamond_r$ replaced by $\overline{\Diamond}_l$ and $\overline{\Diamond}_r$, respectively.*

As for ITL we can derive "proper" I- and E-rules for the two modalities of NL within the SIL system. Below we show the case of $\Diamond_r$:

$$\frac{(j,k):\phi \quad (j,k):fwd \quad (i,j):fwd}{(i,j):\overline{\Diamond}_r\phi}\ \overline{\Diamond}_r I$$

$$\frac{(i,j):\overline{\Diamond}_r\phi \quad (i,j):fwd \qquad \begin{array}{c}[(j,k):\psi]\ [(j,k):fwd] \\ \vdots \\ (m,n):\chi\end{array}}{(m,n):\chi}\ \overline{\Diamond}_r E \quad .$$

The case for $\Diamond_l$ is similar.

CHAPTER 7

# Isabelle

Isabelle [Pau86, Pau89, Pau90, Pau94, Isa01] is a generic proof assistant (written in SML) that supports reasoning in *object-logics* by encoding them (as natural deduction systems) in Isabelle's *meta-logic*. Isabelle is an example of a *logical framework* in the sense of [Pfe01]. The meta-logic of Isabelle is a fragment of intuitionistic higher-order logic[1] [And01, And86] including implication, universal quantification and equality.

Isabelle has been used to formalize and mechanize a wide variety of concepts, from verification of cryptographic protocols [Pau98, BP01] to formalized mathematics [PG96, Fle00, Ras00, Ras01b].

We will in this chapter not give a thorough survey of Isabelle and its use, but concentrate on outlining some of the main principles and ideas. The goal is to make it possible, for someone not familiar with Isabelle, to appreciate the developments and discussions in the subsequent chapters. For a detailed discussion we refer to the comprehensive documentation distributed with Isabelle [Isa01].

## 7.1 Formalizing Logics

In this section we will give a brief overview of how object-logics are defined and encoded in Isabelle.

As mentioned above, the meta-logic of Isabelle is a fragment of intuitionistic higher-order logic including implication $\Longrightarrow$, universal quantification $\bigwedge$ and equality $\equiv$. There is a built-in type *prop* of meta-level truth values; meta-level formulas will have this type. The types (written in Isabelle syntax) of the built-in connectives are

[1] Today, *higher-order logic* and *(simple) type theory* is taken to mean the same [And01].

thus ($\alpha$ is an (almost) arbitrary type):

$$\begin{array}{rcl} \Longrightarrow & :: & [prop, prop] \Rightarrow prop \\ \bigwedge & :: & (\alpha \Rightarrow prop) \Rightarrow prop \\ \equiv & :: & [\alpha, \alpha] \Rightarrow prop \end{array}$$

The type of, say, $\Longrightarrow$ is a function taking two elements of type *prop* and returning an element of type *prop*.

The properties of the meta-logic are defined by a collection of inference rules, including equational rules for the $\lambda$-calculus as well as logical rules.

Isabelle represents logical syntax using the simply typed $\lambda$-calculus: Given the logic in question, a type for each syntactic category and a constant for each symbol is declared.

We will exemplify the basics of Isabelle by considering encoding FOL as an object-logic: FOL is built from two syntactic categories; formulas and terms. In Isabelle we thus define a type $o$ for formulas and a type $D$ for terms.

To establish the connection between the object- and meta-levels we define a coercion constant:

$$Trueprop \quad :: \quad o \Rightarrow prop$$

which is used as a judgment to state when an object-level formula is true on the meta-level.

The constants for the symbols of FOL can now be declared with types as follows:

$$\begin{array}{rcl} \text{true}, \text{false} & :: & o \\ \neg & :: & o \Rightarrow o \\ \wedge, \vee, \rightarrow, \leftrightarrow & :: & [o, o] \Rightarrow o \\ = & :: & [D, D] \Rightarrow o \\ \forall, \exists & :: & (D \Rightarrow o) \Rightarrow o \end{array}$$

The quantifiers are represented as higher-order functions.

Actually, Isabelle has a much more flexible type system than indicated here. Polymorphism is supported such that, e.g., equality could be defined over (almost) arbitrary types. This is managed by means of the concepts of *classes* and *sorts*.

We now turn to the question of formalizing the natural deduction system for FOL. Natural deduction rules for an object-logic are defined by meta-level axioms in Isabelle. Using the *Trueprop* coercion, this allows the meta-logic connectives, $\Longrightarrow$, $\bigwedge$ and $\equiv$ to be read as entailment, variable binding and definitional equality, respectively.

Consider, e.g., the natural deduction rule $\wedge I$ (cf. Section 2.1.5). This rule can be defined by the following meta-level axiom: [2]

$$\bigwedge P.\ \bigwedge Q.\ Trueprop(P) \Longrightarrow (Trueprop(Q) \Longrightarrow Trueprop(P \wedge Q))$$

[2] In connection with Isabelle, we will use $P$ for $\phi/\alpha$, $Q$ for $\psi/\beta$ and $R$ for $\chi/\gamma$.

Such axioms can be notationally simplified: Outermost quantifiers can be dropped, the *Trueprop* coercion made implicit and nested implications rewritten as a list (within brackets $[\![\ldots]\!]$) of premises seperated by semicolons. In other words, the axiom can be stated as follows:

$$[\![P; Q]\!] \Longrightarrow P \wedge Q$$

Here the correspondence to the natural deduction rule is very clear. It is essentially just a matter of syntax.

We now consider the quantifier rule $\forall I$. This rule was formulated as follows in Section 2.2.5:

$$\frac{\phi}{(\forall x)\phi}\ \forall I\ ,$$

with the sidecondition that $x$ is not free in any assumption on which $\phi$ depends. In Isabelle, $\forall I$ can be expressed as follows:

$$\left(\bigwedge x.P(x)\right) \Longrightarrow \forall x.P(x)$$

Here, no sidecondition is stated explicitly. The freeness constraint on $x$ is taken care of by the universal meta-quantifier which, informally speaking, states that $P(x)$ really must hold for all $x$ for the conclusion to be sound.

We now consider how the above concepts are formulated concretely in a way which makes sense to Isabelle. We do this by showing the actual *theory file* which defines the FOL encoding. A theory file defines an object-logic by extending Isabelle's meta-logic theory (`Pure`).

We first show the declaration of the necessary types and constants in the ASCII syntax used by Isabelle. Note how some operators are declared to be infix (with left or right association).

```
FOL = Pure +
types
  o D
consts
  Trueprop      :: o => prop

  True          :: o
  False         :: o
  Not           :: "o => o"               ("~ _" [40] 40)
  "&"           :: "[o, o] => o"          (infixr 35)
  "|"           :: "[o, o] => o"          (infixr 30)
  "-->"         :: "[o, o] => o"          (infixr 25)
  "<->"         :: "[o, o] => o"          (infixr 25)
  All           :: "(D => o) => o"        (binder "ALL " 10)
  Ex            :: "(D => o) => o"        (binder "EX " 10)
  "="           :: "[D, D] => o"          (infixl 50)
```

Now for the declarations of the actual natural deduction rules. Note that $[\![\ldots]\!]$ is written `[|...|]`, $\Longrightarrow$ is written `==>`, $\bigwedge$ is written `!!` and $\equiv$ is written `==`.

```
rules
  conjI         "[| P; Q |] ==> P&Q"
  conjE1        "P&Q ==> P"
  conjE2        "P&Q ==> Q"

  disjI1        "P ==> P|Q"
  disjI2        "Q ==> P|Q"
  disjE         "[| P|Q; P ==> R; Q ==> R |] ==> R"

  impI          "(P ==> Q) ==> P-->Q"
  mp            "[| P-->Q;  P |] ==> Q"

  FalseE        "(P-->False ==> False) ==> P"

  refl          "s=s"
  subst         "[| s=t; P(s) |] ==> P(t)"

  allI          "(!!x. P(x)) ==> (ALL x. P(x))"
  allE          "(ALL x. P(x)) ==> P(s)"
  exI           "P(s) ==> (EX x. P(x))"
  exE           "[| EX x. P(x); !!x. P(x) ==> R |] ==> R"

defs
  True_def      "True    == False-->False"
  not_def       "~P      == P-->False"
  iff_def       "P<->Q   == (P-->Q) & (Q-->P)"
end
```

Natural deduction rules are not stated for true, $\neg$ and $\leftrightarrow$ since these connectives are defined in terms of other connectives. It is easy to derive proper I- and E-rules for these defined connectives.

### 7.1.1 Sequent Calculus

Isabelle expects object-logics to be encoded in a natural deduction style. But it is possible to encode a sequent based logic in Isabelle. Informally speaking, sequent rules can be regarded as natural deduction rules over sequents instead of "plain" formulas. This means that by defining a type of sequents in Isabelle, the sequent rules can be declared as meta-level axioms as above, where *Trueprop* now maps sequents (instead of formulas) to *prop*. The sequents can themselves be defined as pairs of sequences of formulas (of type *o*). In a concrete representation in Isabelle, it is not feasible to use sequents of multisets, hence sequences are used.

The Isabelle distribution includes a sequent theory which provides the basic notions needed for defining a sequent based logic. We refer to the documentation for a

detailed discussion of how this is done [Pau01a].

We will here briefly consider the concrete syntax used for sequent based logics. The sequent $\Gamma, P, Q, \Delta \vdash R$ is in Isabelle represented by:

```
$G, P, Q, $D |- R
```

In other words, variables representing an arbitrary sequence are designated by prefixing a $.

As an example, we below give the Isabelle definitions of the sequent rules for PL:

```
rules
  basic "$H, P, $G |- $E, P, $F"

  conjR "[| $H |- $E, P, $F; $H |- $E, Q, $F |] ==> $H|- $E, P&Q, $F"
  conjL "$H, P, Q, $G |- $E ==> $H, P&Q, $G |- $E"

  disjR "$H |- $E, P, Q, $F ==> $H |- $E, P|Q, $F"
  disjL "[| $H, P, $G |- $E; $H, Q, $G |- $E |] ==> $H, P|Q, $G |- $E"

  impR  "$H, P |- $E, Q, $F ==> $H |- $E, P-->Q, $F"
  impL  "[| $H, $G |- $E, P; $H, Q, $G |- $E |] ==> $H, P-->Q, $G |- $E"

  notR  "$H, P |- $E, $F ==> $H |- $E, ~P, $F"
  notL  "$H, $G |- $E, P ==> $H, ~P, $G |- $E"

  FalseL "$H, False, $G |- $E"
```

These rules correspond very closely to the rules stated in Section 2.1.4. The only difference is that, e.g., (L$\wedge$) is stated as:

$$\frac{\Gamma, \alpha, \beta, \Gamma' \vdash \Delta}{\Gamma, \alpha \wedge \beta, \Gamma' \vdash \Delta}\ (\mathrm{L}\wedge) \quad .$$

This is to make reasoning less tedious as the sequents are based on sequences instead of multisets.

In the actual encoding, structural rules, quantifier rules, etc., are included as well.

## 7.2 Constructing Proofs

Above we sketched how object-logics can be encoded in Isabelle. But how do we conduct proofs in such encodings?

We will in this section consider proof construction in Isabelle. We will concentrate on backwards proof construction. The central concept here is resolution: In a backwards proof, a goal is unified with the conclusion of a rule whose premises become new subgoals.

A *proof state* in Isabelle is a meta-formula of the following form:

$$[\![\phi_1, \ldots, \phi_n]\!] \Longrightarrow \phi \ .$$

To prove the formula $\phi$, take $\phi \Longrightarrow \phi$ as the initial proof state. This is trivially a meta-theorem. Now iteratively refine the proof state by performing resolution with suitable rules. At some point during the proof, a typically proof state is $[\![\phi_1, \ldots, \phi_n]\!] \Longrightarrow \phi$. This proof state is a theorem stating that the subgoals $\phi_1, \ldots, \phi_n$ imply $\phi$. When at some point $n = 0$ we have proven $\phi$ to be a theorem of the object logic in question.

The actual (backwards) proof construction is guided by *tactics* and *tacticals*. An Isabelle tactic is a function taking a proof state and returning a sequence of possible successor states. The basic and most important tactics are the standard resolution tactic (`resolve_tac`) which takes a list of rules, (tries to) unify the conclusion of the selected subgoal with the conclusions of the rules and returns states for each combination of premises of the rules and unifier, and the assumption tactic (`assume_tac`) which (tries to) unify the conclusion of the selected subgoal with its assumptions and returns states for each combination of assumptions and unifier.

Below we illustrate how this works in Isabelle by proving $P \wedge Q \rightarrow P$ in the `FOL` theory. The formula to be proven is submitted to Isabelle by means of the `Goal` function. Isabelle responds by giving the formula to be proven and a list of the subgoals needed to establish it (in this case just one):

```
ML> Goal "P & Q --> P";
P & Q --> P
 1. P & Q --> P
```

By using the resolve tactic twice and the assumption tactic once, the goal is proven.

```
ML> by (resolve_tac [impI] 1);
P & Q --> P
 1. P & Q ==> P

ML> by (resolve_tac [conjE1] 1);
P & Q --> P
 1. P & Q ==> P & ?Q1

ML> by (assume_tac 1);
P & Q --> P
No subgoals!
```

Isabelle uses *schematic variables* (which are prefixed by a ?, such as `?Q1` in the above extract) for unknowns that can later in the course of a proof be instantiated. This happens in the above example when `assume_tac` instantiates `?Q1` to `Q` to prove the goal by assumption.

There are many other (and more sophisticated) tactics and it is possible to write ones own. This can, e.g., be done by means of tacticals, which are higher-order

functions used to combine basic tactics. Such combinations of tactics can be from simple sequential compositions to more advanced cases of special-purpose search tactics.

## 7.3 The Classical Reasoner

Isabelle comes with a so-called *classical reasoner* which provides a range of tactics performing various kinds of proof searches. The classical reasoner is *generic* in the sense that it can be instantiated and used for a large class of object-logics — in essence, any object-logic which is based on/includes classical propositional logic formulated in a natural deduction style.

One of the tactics provided is `Fast_tac` which performs a simple depth-first search. Using this, our example above can be solved in one go:

```
ML> Goal "P & Q --> P";
P & Q --> P
 1. P & Q --> P

ML> by (Fast_tac 1);
P & Q --> P
No subgoals!
```

The power of the classical reasoner is due to its generality and expandability, such that not only rules for the classical connectives are considered in a search. Consider, e.g., defining set theory in Isabelle: The natural deduction rules for unions and intersections resemble those for disjunction and conjunction. By adding these to the classical reasoner, effective reasoning for set theory is achieved. The following chapters will also illustrate why the expandability of the classical reasoner is very convenient and useful.

For details on the inner workings of the classical reasoner we refer to [Pau01b]. The classical reasoner of Isabelle is an incarnation of the idea of generic automatic proof tools as discussed in [Pau97].

## 7.4 The Simplifier

Rewriting (e.g., [BN98]) in Isabelle is based on the theory of ordered rewriting [MN90] and is handled by Isabelle's *simplifier* [Pau01b].

Rewriting is done on the meta-logic level with respect to $\equiv$. It is therefore necessary to tell Isabelle when an object-logic equality/equivalence is a meta-level equality. This is in the case of FOL done by adding the following reflection rules to the theory file:

```
eq_reflection:  "(x=y)   ==> (x==y)"
iff_reflection: "(P<->Q) ==> (P==Q)"
```

The rewrite rules used by the simplifier come from three sources:

1. Occurring in the (permanent) *simplification set*.
2. Explicitly stated by the user.
3. Extracted from the list of assumptions of a goal.

The four main simplification tactics are:

```
Simp_tac
Asm_simp_tac
Full_simp_tac
Asm_full_simp_tac
```

The prefix `Asm` means that additional rewrite rules should be extracted from the assumptions (3. above). The prefix `Full` means that the simplifier should simplify the assumptions themselves as well.

Setting up the simplifier for FOL includes proving a lot of (trivial) rules which are then added to the simplification set. Such rules include:

```
"P & True <-> P"
"P & P <-> P"
"P & P & Q <-> P & Q",
"P & ~P <-> False"
"P | False <-> P"
"P | P <-> P",
"~ False <-> True"
"(False --> P) <-> True"
```

Furthermore, the *solver* (which is part of the simplifier) is set up to prove trivial goals resulting from simplification.

Setting up the simplifier for an object-logic is in general a non-trivial task. Many details and advanced features have not been mentioned here.

Chapter 8

# Interval Logics in Isabelle

In the previous chapter we gave a brief, general introduction to Isabelle. In this chapter we will consider encodings in Isabelle of the proof systems for SIL discussed in Chapters 5 and 6.

We begin in Section 8.1 by considering an encoding of the LND system for SIL. This includes a discussion on how we have set up and instantiated the simplifier and the classical reasoner. Furthermore, we consider encodings of the extensions to the LND system discussed in Section 6.2 as well as the use of the encoding as a general framework (cf. Section 6.3).

In Section 8.2 we consider an encoding of the sequent calculus system for SIL. Again, we discuss both the simplifier and the reasoner. Finally, in Section 8.3 we consider related work. This includes both modal and interval logic encodings.

If the reader several places notices great similarities between how the theory was formulated and how the encoding is done, it is not coincidental. The theory was deliberately formulated with an Isabelle encoding in mind, as well as the encoding itself was kept as close to the theory as possible.

## 8.1 Labelled Natural Deduction

In this section we consider an encoding of the LND system for SIL in Isabelle.

### 8.1.1 Encoding the LND system

The encoding of the LND system is a modification and extension of the "pure" FOL encoding described in Section 7.1. We will refer to the encoding described in this

section as Isabelle/LSIL.

First, we need to define an additional type for elements of the temporal domain:

```
types
  T
```

The basic judgment of FOL, *Trueprop*, is then discarded in favor of a judgment for labelled formulas:

```
consts
  LF            :: "[T, T, o] => prop"     ("(<_,_> : (_))" [6,6,5] 4)
```

A concrete syntax is defined for the judgment such that `<i,j>:P` is used for $(i, j) : \phi$.[1]

We also define the auxiliary judgments ri and cf:

```
  RI            ::  'a::logic => prop      ("(RI _)")
  CF            ::          o => prop      ("(CF _)")
```

Note here how the polymorphism of Isabelle is utilized such that `RI` can be used for both terms and formulas.

We now turn to the definition of connectives and symbols. The first order operators and quantifiers are defined exactly as in the `FOL` theory. We additionally define the chop modality as `^` and the $\ell$ symbol as `len`, as well as the symbols used for the duration domain:

```
  True          :: o
  False         :: o
  Not           :: "o => o"                ("~ _" [40] 40)
  "&"           :: "[o, o] => o"           (infixr 35)
  "|"           :: "[o, o] => o"           (infixr 30)
  "-->"         :: "[o, o] => o"           (infixr 25)
  "<->"         :: "[o, o] => o"           (infixr 25)
  All           :: "(D => o) => o"         (binder "ALL " 10)
  Ex            :: "(D => o) => o"         (binder "EX " 10)
  "="           :: "[D, D] => o"           (infixl 50)

  "^"           :: "[o, o] => o"           (infixr 38)
  len           :: D
  null          :: D                       ("0")
  one           :: D                       ("1")
  "-"           :: D => D                  ("- _" [80] 80)
  "+"           :: [D, D] => D             (infixl 60)
```

We have now come to the LND rules. The propositional rules are encoded as for plain FOL (cf. Section 7.1) but with the addition of labels:

[1] Remember, in connection with Isabelle we use $P$ for $\phi/\alpha$, $Q$ for $\psi/\beta$ and $R$ for $\chi/\gamma$.

```
rules
  conjI    "[| <i,j>:P;  <i,j>:Q |] ==> <i,j>:P&Q"
  conjE1   "<i,j>:P&Q ==> <i,j>:P"
  conjE2   "<i,j>:P&Q ==> <i,j>:Q"
  disjI1   "<i,j>:P ==> <i,j>:P|Q"
  disjI2   "<i,j>:Q ==> <i,j>:P|Q"
  disjE    "[| <i,j>:P|Q; <i,j>:P ==> <k,l>:R; <i,j>:Q ==> <k,l>:R |]
              ==> <k,l>:R"
  impI     "(<i,j>:P ==> <i,j>:Q) ==> <i,j>:P-->Q"
  mp       "[| <i,j>:P-->Q;  <i,j>:P |] ==> <i,j>:Q"
  FalseE   "(<i,j>:P-->False ==> <k,l>:False) ==> <i,j>:P"
```

The remaining LND rules are defined straightforwardly in the Isabelle syntax based on the definitions of Section 6.1:

```
  chopI    "[| <i,k>:P; <k,j>:Q |] ==> <i,j>:P^Q"
  chopE    "[| <i,j>:P^Q; !!k. [| <i,k>:P; <k,j>:Q |] ==> <m,n>:R |]
              ==> <m,n>:R"

  uniq1    "[| RI s; <i,k>:len=s; <i,l>:len=s; <l,j>:P |] ==> <k,j>:P"
  uniq2    "[| RI s; <k,j>:len=s; <l,j>:len=s; <i,l>:P |] ==> <i,k>:P"
  zero     "<i,i>:len=0"
  plusI    "[| <i,k>:len=s; <k,j>:len=t; RI s; RI t |]
              ==> <i,j>:len=s+t"
  plusE    "[| <i,j>:len=s+t; RI s; RI t;
              !!k. [| <i,k>:len=s; <k,j>:len=t |] ==> <m,n>:R |]
              ==> <m,n>:R"

  refl     "<i,j>:s=s"
  subst    "[| CF P(x); <i,j>:s=t; <i,j>:P(s) |] ==> <i,j>:P(t)"
  substRI  "[| RI s; RI t; <i,j>:s=t; <i,j>:P(s) |] ==> <i,j>:P(t)"

  allI     "(!!x. RI x ==> <i,j>:P(x)) ==> <i,j>:(ALL x. P(x))"
  allERI   "[| RI s; <i,j>:(ALL x. P(x)) |] ==> <i,j>:P(s)"
  allECF   "[| CF P(x); <i,j>:(ALL x. P(x)) |] ==> <i,j>:P(s)"
  exIRI    "[| RI s; <i,j>:P(s) |] ==> <i,j>:(EX x. P(x))"
  exICF    "[| CF P(x); <i,j>:P(s) |] ==> <i,j>:(EX x. P(x))"
  exE      "[| <i,j>:EX x. P(x); !!x. [| RI x; <i,j>:P(x) |]
              ==> <m,n>:R |] ==> <m,n>:R"

  com      "<i,j>:s + t = t + s"
  ass      "<i,j>:(s + t) + u = s + (t + u)"
  idn      "<i,j>:s + 0 = s"
  inv      "<i,j>:s + -s = 0"
```

It is here worth noticing how the sideconditions concerning the non-occurrence of worlds in assumptions (as for $\frown E$) are handled similarly to the sideconditions concerning freeness of variables in assumptions (as for $\exists E$). In Section 6.1.2, the rule

$\frown E$ is stated as follows:

$$\dfrac{\begin{array}{cc}(i,j):\phi\frown\psi & \begin{array}{c}[(i,k):\phi]\ [(k,j):\psi]\\ \vdots\\ (m,n):\chi\end{array}\end{array}}{(m,n):\chi}\ \frown E\ ,$$

with the sidecondition that $k$ is different from $i, j, n, m$, and does not occur in any assumption on which the upper occurrence of $(m,n):\chi$ depends except $(i,k):\phi$ and $(k,j):\psi$. This sidecondition is, by means of meta-quantification, handled similarly to how freeness of variables is handled as discussed in Section 7.1; thus, no sideconditions are stated explicitly in either case. Using the same construct in both cases is possible due to the higher-order meta-logic of Isabelle.

All we need now to complete the basic SIL encoding are the rules for the ri and cf judgments; below we give a selection of these rules. It should be obvious how the rules for the remaining cases are defined (cf. Section 6.1.1 where the rules are listed).

```
rigid      "[| <k,l>:P; RI P |] ==> <i,j>:P"

RIfalse    "RI False"
RIconjI    "[| RI P; RI Q |] ==> RI (P&Q)"
RIconjE1   "RI (P&Q) ==> RI P"
RIconjE2   "RI (P&Q) ==> RI Q"
RIallI     "(RI s ==> RI P(s)) ==> RI (ALL x. P(x))"
RIallE     "[| RI s; RI (ALL x. P(x)) |] ==> RI P(s)"
RIequI     "[| RI s; RI t |] ==> RI (s=t)"
RIequE1    "RI (s=t) ==> RI s"
RIequE2    "RI (s=t) ==> RI t"
RInegI     "RI s ==>  RI (-s)"
RInegE     "RI (-s) ==> RI s"
RIzero     "RI 0"

CFfalse    "CF False"
CFimpI     "[| CF P; CF Q |] ==> CF (P-->Q)"
CFimpE1    "CF (P-->Q) ==> CF P"
CFimpE2    "CF (P-->Q) ==> CF Q"
CFexI      "CF P(s) ==> CF (EX x. P(x))"
CFexE      "CF (EX x. P(x)) ==> CF P(s)"
CFequ      "CF (s=t)"
```

### 8.1.2 Simplification

In this section we discuss how we have set up and instantiated the simplifier for Isabelle/LSIL.

### Labelled Simplification

In Section 7.4 we saw how we in the theory file must define coercions stating that certain object-level equalities are meta-level equivalent as well, for making it possible to use the simplifier.

We are now in a labelled formalism and a naive modification to those basic coercion assertions would not give the desired result. Consider:

```
eq_reflection   "<i,j>:s=t ==> (s==t)"
iff_reflection  "<i,j>:P<->Q ==> (P==Q)"
```

This is unsound. Because $s = t$ on a specific interval does not mean that $s$ can be replaced by $t$ unconditionally everywhere — that is what $s \equiv t$ would imply.

What we need are rules saying that if it is actually the case that $s = t$ on all intervals then in fact $s \equiv t$ is sound. This can be expressed as follows:

```
eq_reflection   "(!!i j. <i,j>:s=t) ==> (s==t)"
iff_reflection  "(!!i j. <i,j>:P<->Q) ==> (P==Q)"
```

This is sound and help us a long way. In particular, all theorems with no assumptions can be added directly to the simplifier by means of these reflection rules.

But the approach also has its limitations. This is, e.g., the case if we want to use rewrite rules of the assumptions of a goal. As an example, consider:

```
Goal "[| <k,l>:len=a; <i,j>:len=b |] ==> <i,j>:len+a=b+a";
```

Here we would like `len` in the conclusion of the goal to be rewritten to `b`. This is not possible with the above reflection rules alone. As we are only interested in the particular interval $(i, j)$, it is useful to know that `len=b` holds on that interval even if it does not hold on all intervals.

A solution is to "lift" labels to the meta-level during rewriting. [2] For this to work we have to define an additional judgment, which, informally speaking, is used to state which interval we are currently "on":

```
ON      :: "[T, T]    => prop"     ("(<_,_>)")
```

This judgment is put to use in the following two rules (and nowhere else):

```
on_eq_reflection  "[| <i,j>:s=t; <i,j> |] ==> (s==t)"
on_cong           "(<i,j> ==> P == P') ==> <i,j>:P == <i,j>:P'"
```

The last rule is a congruence rule (such rules are also handled by the simplifier) saying that the labelled formulas $(i, j) : \phi$ and $(i, j) : \phi'$ are equivalent if $\phi$ and $\phi'$ can be shown equivalent under the assumption that we are currently on the interval $(i, j)$. Together with the first rule, which states that $s$ and $t$ are equivalent if $s = t$

[2] We would like to thank Sebastian Skalberg for coming up with the idea which lead to this solution.

on the interval $(i, j)$ and that we in fact currently are on the interval $(i, j)$, these two rules achieve what we want.

In a broader perspective, the above way of handling labelled simplification in Isabelle is not only useful in the cases of labels being intervals. The principles could also be applied in connection with the modal logic encodings in Isabelle of [Vig00] and for other labelled logics in general as well.

### Simplification for Abelian Groups

The (signed duration) domain of SIL has in its basic form the structure of an Abelian group. The simplifier can help ease reasoning over this domain by making it possible to reduce any term to its unique normal form automatically.

The simplifier of Isabelle is based on the theory of ordered rewriting [MN90]. A complete set of reductions [BN98] for Abelian groups exists within this theory; they are as follows:

$$\begin{array}{rclcrcl} s+t & = & t+s, & & s+(t+u) & = & t+(s+u), \\ (s+t)+u & = & s+(t+u), & & s+-s & = & 0, \\ 0+s & = & s, & & s+0 & = & s, \\ s+(-s+t) & = & t, & & -(s+t) & = & -s+-t, \\ -(-s) & = & s, & & -0 & = & 0. \end{array}$$

These equalities can all be proved in Isabelle given the four basic axioms defining an Abelian group.

In [MN90] it is shown that the above ten rewrite rules give a ground complete rewrite system[3] if the constants $a_1, a_2, \ldots$ are strictly ordered as follows: $a_1 < -a_1 < a_2 < -a_2 < \cdots$. Such an ordering can be achieved by a lexicographic path ordering [BN98] with the following precedence of the symbols: $+ < - < a_1 < a_2 < \cdots$. For this we keep Isabelle's standard strict ordering $a_1 < a_2 < \ldots$ on constants and extend it to $+ < - < a_1 < a_2 < \ldots$ as follows:

```
fun ord (Const("op +", _), Const("op +", _)) = EQUAL
  | ord (_,                Const("op +", _)) = GREATER
  | ord (Const("op +", _), _                ) = LESS
  | ord (Const("-", _),    Const("-", _))    = EQUAL
  | ord (_,                Const("-", _))    = GREATER
  | ord (Const("-", _),    _             )   = LESS
  | ord (f,g) = hd_ord (f,g);
```

where `hd_ord` is Isabelle's standard ordering on constants.

The standard lexicographic ordering on terms used in Isabelle is not a path ordering and we thus have to redefine it; the lexicographic path ordering induced by `ord` can be defined as follows [BN98]:

[3] A system where any ground term can be rewritten to its unique normal form.

```
fun lpo (Abs (_, T, t), Abs(_, U, u)) =
      (case lpo (t, u) of EQUAL => typ_ord (T, U) | ord => ord)
  | lpo (t, u) =
      let val (f, ts) = strip_comb t and (g, us) = strip_comb u in
        if forall (fn ti => lpo (ti,u) = LESS) ts
        then case ord(f,g) of
               GREATER => if forall (fn ui => lpo (t,ui) = GREATER) us
                          then GREATER else LESS
             | EQUAL   => if forall (fn ui => lpo (t,ui) = GREATER) us
                          then list_ord lpo (ts,us) else LESS
             | LESS    => LESS
        else GREATER
      end

fun ag_termless (t,u) = (lpo (u,t) = GREATER);
```

Note that we do not have variables explicitly present in the ordering as they are regarded as special constants in Isabelle.

### Setting up the Simplifier

The simplifier is set up as indicated above by modifying it for use for labelled simplification (which among other things means adding the congruence rule concerning the `ON` judgment to the simplification set). Furthermore, the complete set of rewrite rules for Abelian groups is added to the standard simplification set and, importantly, the correct term ordering (`ag_termless`) is set.

Utilizing the rules defining the ri/cf judgments, (conditional) meta rewrite rules can be proved:

```
           "(RI ~P) == RI P";
"RI P ==> (RI P&Q) == RI Q";
"RI P ==> (RI P^Q) == RI Q";
"RI s ==> (RI s=t) == RI t";

           "(CF ~P) == CF P";
"CF P ==> (CF P&Q) == CF Q";
```

and so on for all remaining connectives/symbols. These rewrite rules are all added to the standard simplification set.

Finally, standard FOL rewrite rules (as mentioned in Section 7.4) are proved in the labelled system and added to the simplification set as well.

There are more (technical) details to the actual setting up of the simplifier — we have here merely given a overview.

### 8.1.3 Extensions to Labelled SIL

In this section we consider how the extensions to the LND system for SIL, as discussed in Section 6.2, have been encoded in Isabelle.

#### Totally Ordered Infinite Field

In the case of the total order we first have to define the appropriate binary relations:

```
"<="          :: "[D, D] => o"                (infixl 50)
"<"           :: "[D, D] => o"                (infixl 50)
less_def      "s < t == (s <= t & s ~= t)"
```

The necessary axioms for the total order can now be stated:

```
orefl         "<i,j>:s<=s"
otran         "[| <i,j>:s<=t; <i,j>:t<=u |] ==> <i,j>:s<=u"
oanti         "[| <i,j>:s<=t; <i,j>:t<=s |] ==> <i,j>:s=t"
oline         "<i,j>:s<=t | t<=s"
ocong         "<i,j>:s<=t ==> <i,j>:s+u<=t+u"
```

The Archimedian axioms are straightforward:

```
oinfd         "<i,j>:EX s. s < t"
oinfu         "<i,j>:EX s. t < s"
```

Finally, the extension to a field is handled by the below definitions:

```
"`"           :: D => D              ("(_`)" [90] 90)
"*"           :: [D, D] => D         (infixl 70)

zlo           "<i,j>:0 < 1"

mcom          "<i,j>:s * t = t * s"
mass          "<i,j>:(s * t) * u = s * (t * u)"
midn          "<i,j>:s * 1 = s"
minv          "<i,j>:s~=0 ==> <i,j>:s * s` = 1"
mdis          "<i,j>:s * (t + u) = s * t + s * u"

mocong        "[| <i,j>:0<u; <i,j>:s<=t |] ==> <i,j>:s*u<=t*u"
iocong        "<i,j>:0<s ==> <i,j>:0<s`"
```

Note how we have to explicitly state that $0 < 1$.

#### Signed Duration Calculus

To encode SDC we have to define a type for state expressions as well as the duration-function $\int$ mapping state formulas to elements of the duration domain:

```
types
  s
consts
  dur           :: "s => D"
```

The (Boolean) operators occurring in state formulas are defined straightforwardly:

```
  TOP           :: s
  BOT           :: s
  NOT           :: "s => s"                     ("NOT _" [25] 25)
  AND           :: "[s, s] => s"                (infixr 20)
  OR            :: "[s, s] => s"                (infixr 15)
  IMP           :: "[s, s] => s"                (infixr 10)
  IFF           :: "[s, s] => s"                (infixr 10)
```

The abbreviation $\lceil S \rceil$ is defined as well:

```
  high          :: "s => o"                     ("(‘_‘)")
  high_def      "‘S‘ == dur(S) = len & len ~= 0"
```

We have now come to the SDC axioms and rules. They are basically defined as discussed in Section 6.2.2:

```
DA1       "<i,j>:dur(BOT) = 0"
DA2       "<i,j>:dur(TOP) = len"
DA3a      "<i,j>:len <= 0 ==> <i,j>:dur(S) <= 0"
DA3b      "<i,j>:0 <= len ==> <i,j>:0 <= dur(S)"
DA4       "<i,j>:dur(S1) + dur(S2) = dur(S1 OR S2) + dur(S1 AND S2)"
DA5       "[| <i,k>:dur(S)=s; <k,j>:dur(S)=t; RI s; RI t |]
          ==> <i,j>:dur(S)=s+t"
DA6       "<k,l>:ST (S1 IFF S2) ==> <i,j>:dur(S1) = dur(S2)"

IRr       "[| <i,j>:len=0 --> P; <i,j>:P|^|‘S‘ --> P;
              <i,j>:P|^|‘NOT S‘ --> P |] ==> <i,j>:P"
IRl       "[| <i,j>:len=0 --> P; <i,j>:‘S‘|^|P --> P;
              <i,j>:‘NOT S‘|^|P --> P |] ==> <i,j>:P"
```

The definition of the axiom DA6 requires some explanation. In Section 4.3.1 it is formulated as follows:

DA6: $\int S_1 = \int S_2$ if $S_1 \leftrightarrow S_2$ in propositional logic.

Hence, the sidecondition has to be formulated an appropriate way in Isabelle.

One approach would be to define a complete propositional logic over the state type `s` from scratch. But this would be somewhat cumbersome as we already have a propositional logic as part of our main SIL logic. What we choose to do is to define a coercion `ST` from `s` to `o`, together with appropriate equivalences:

```
ST      :: "s => o"        ("(ST _)")

STTOP   "ST TOP == True"
STBOT   "ST BOT == False"
STNOT   "ST (NOT P) == ~(ST P)"
STAND   "ST (P AND Q) == (ST P) & (ST Q)"
STOR    "ST (P OR Q) == (ST P) | (ST Q)"
STIMP   "ST (P IMP Q) == (ST P) --> (ST Q)"
STIFF   "ST (P IFF Q) == (ST P) <-> (ST Q)"
```

If these equivalences are added to the simplifier, we can, after having resolved with DA6, automatically convert a state formula to a propositional SIL formula which then is decidable by means of the classical reasoner. This process can be encapsulated completely in a tactic, thus making the coercion `ST` invisible to the user.

For further convenience, we add rewrite rules to the simplifier such as the following:

```
"<i,j>:dur(Q OR (P IMP R)) = dur(P IMP Q OR R)"
"<i,j>:dur(NOT (P AND Q)) = dur(NOT P OR NOT Q)"

"<i,j>:‘P OR P OR Q‘ <-> ‘P OR Q‘"
"<i,j>:‘(P OR Q) OR R‘ <-> ‘P OR (Q OR R)‘"
```

### 8.1.4 Isabelle/LSIL as a General Framework

In this section we consider the encoding of ITL and NL on top of Isabelle/LSIL. The theoretical aspects of this were discussed in Section 6.3.2.

#### ITL

We first need to define the abbreviated modality $\frown$:

```
"|^|"          :: "[o, o] => o"       (infixr 38)
chopsub_def    "P|^|Q == (P & fwd)^(Q & fwd)"
```

I- and E-rules can now be derived in Isabelle: [4]

```
Goal "[| <i,k>:fwd; <k,j>:fwd; <i,k>:P; <k,j>:Q |] ==> <i,j>:P|^|Q";

Goal "[| <i,j>:P|^|Q; <i,j>:fwd; !!k. [| <i,k>:P; <k,j>:Q;
         <i,k>:fwd; <k,j>:fwd|] ==> <l,m>:R |] ==> <l,m>:R";
```

[4] To emphasize that a rule is derived — in contrast to being stated as an axiom — we will show the `Goal` command submitted to Isabelle. We will leave out the actual proof, though. This convention will be used for the rest of the thesis.

**NL**

First the definitions of the abbreviated modalities $\overline{\diamond}_r$ and $\overline{\diamond}_l$:

```
srn        :: o => o    ("<R>_" [50] 50)
sln        :: o => o    ("<L>_" [50] 50)
srn_def    "<R>P == True^(len=0 & (P^bwd))"
sln_def    "<L>P == (len=0 & (bwd^P))^True"
```

Then the derived I-/E-rules:

```
Goal "[|<i,j>:fwd; <j, k>: fwd; <j,k>: P |] ==> <i,j>: <R>P";

Goal "[| <i,j>:<R>P; <i,j>:fwd; !!k. [| <j,k>: fwd; <j,k>: P |]
         ==> <m,n>:R |] ==> <m,n>:R";

Goal "[|<i,j>:fwd; <k, i>:fwd; <k,i>:P|] ==> <i,j>: <L>P";

Goal "[| <i,j>:<L>P; <i,j>:fwd; !!k. [| <k,i>: fwd; <k,i>: P |]
         ==> <m,n>:R |] ==> <m,n>:R";
```

### 8.1.5 The Classical Reasoner

The generality and expandability of the classical reasoner was discussed in Section 7.3. Since Isabelle/LSIL includes classical propositional logic and is formulated in a natural deduction style, the classical reasoner can be instantiated for it. In doing this (besides the propositional rules) we add I-/E-rules for $\frown$ and $\ell$ as well as the axiom `zero`.

The use of the reasoner is in the spirit of [Pau97]: When additional modalities are defined, say, $\overline{\frown}$ as considered above, the derived I-/E-rules are added directly to the classical reasoner. Reasoning is thus done on a higher level of abstraction as the definitions (almost) never need to be expanded.

## 8.2 Sequent Calculus

In this section we give an overview of the encoding in Isabelle of the sequent calculus for SIL as discussed in Chapter 5.

The basics of defining sequent based logics in Isabelle were considered in Section 7.1.1.

### 8.2.1 Encoding the Sequent Calculus for SIL

The encoding of the sequent calculus for SIL can be based almost directly on the basic FOL sequent calculus encoding since we do not have to introduce any new

types (contrary to the LND encoding). The basic judgment is unchanged too, i.e., it is the coercion mapping sequents to meta-level truth values (cf. Section 7.1.1).

The definition in Isabelle of operators, modalities and symbols of SIL is completely similar to that of the LND system. Additionally, we define the unary modalities $\Box$ and $\Diamond$ as considered in Section 5.1:

```
dia         :: o => o       ("<>_" [50] 50)
box         :: o => o       ("[]_" [50] 50)
dia_def     "<>P == True^P^True"
box_def     "[]P == ~(<>(~P))"
```

In Section 5.1 the sequent rules are formulated in such a way that the sideconditions concerning rigidity and chop-freeness are given on the meta-level. In the Isabelle encoding we have to make these a part of the system itself. We do this in the exact same way as in the case of the LND system, thus we define the two judgments `RI` and `CF`, together with all defining rules, as in Section 8.1.1.

The propositional sequent calculus rules are those of Section 7.1.1. The quantifier rules get the following form (because of the ri/cf judgments):

```
allR  "(!!x. RI x ==> $H |- $E, P(x), $F)
       ==> $H |- $E, ALL x. P(x), $F"

allLRI "[| RI x; $H, P(x), $G, ALL x. P(x) |- $E |]
       ==> $H, ALL x. P(x), $G |- $E"
allLCF "[| CF P(x); $H, P(x), $G, ALL x. P(x) |- $E |]
       ==> $H, ALL x. P(x), $G |- $E"

exRRI  "[| RI x; $H |- $E, P(x), $F, EX x. P(x) |]
       ==> $H |- $E, EX x. P(x), $F"
exRCF  "[| CF P(x); $H |- $E, P(x), $F, EX x. P(x) |]
       ==> $H |- $E, EX x. P(x), $F"

exL   "(!!x. RI x ==> $H, P(x), $G |- $E)
       ==> $H, EX x. P(x), $G |- $E"
```

The encoding of the S4 modal rules (cf. Section 3.4) is not completely straightforward. The special sideconditions embedded in the rules (concerning $\Box$ and $\Diamond$) have to be handled somehow. Our approach to this is inspired by the principles of the (undocumented) modal object-logics distributed with Isabelle. This means that we handle the side conditions of the rules (R$\Box''$), (L$\Diamond''$) and (LRM) by means of a certain set of Horn clauses:

```
lstar0         "|L>"
lstar1         "$G |L> $H ==> []P, $G |L> []P, $H"
lstar2         "$G |L> $H ==>   P, $G |L>      $H"
rstar0         "|R>"
rstar1         "$G |R> $H ==> <>P, $G |R> <>P, $H"
rstar2         "$G |R> $H ==>   P, $G |R>      $H"
```

These Horn clauses define two relations between sequences, where, e.g., `$G |L> $H` iff `$H` only contains formulas of the form `[]P` and all formulas of `$H` also are in `$G`. Similarly for `$G |R> $H` but now for formulas of the form `<>P`. Given this, we can formulate the S4 rules as follows:

```
boxR   "[| $E |L> $E'; $F |R> $F'; $G |R> $G';
           $E' |- $F', P, $G'|] ==> $E |- $F, []P, $G"

boxRRI "[| RI P; $E' |- $F', P, $G'|] ==> $E |- $F, []P, $G"

boxL      "$E, P, $F |- $G ==> $E, []P, $F |- $G"

diaR      "$E |- $F, P, $G ==> $E |- $F, <>P, $G"

diaL   "[| $E |L> $E'; $F |L> $F'; $G |R> $G';
           $E', P, $F' |- $G'|] ==> $E, <>P, $F |- $G"

diaLRI "[| RI P; $E', P, $F' |- $G'|] ==> $E, <>P, $F |- $G"
```

The remaining sequent rules concerning $\frown$ and $\ell$ are defined as expected based on the definitions in Section 5.1.

### 8.2.2 The Simplifier

As the sequent calculus encoding is non-labelled we do not have the problem concerning labelled simplification. But we have a related problem which must be addressed for the simplifier to be more flexible.

In the FOL sequent calculus distributed with Isabelle, the following congruence rule is included:

```
left_cong  "[| P == P';  |- P' ==> ($H |- $F) == ($H' |- $F') |]
            ==> (P, $H |- $F) == (P', $H' |- $F')"
```

This rule is used for extracting rewrite rules from the antecedent of the sequent for use in the succedent. In other words, if this rule is not included, potentiel rewrite rules of the antecedent are unavailable.

Unfortunately, this congruence rule is generally not sound for modal logics. It is sound though, if all formulas of the sequent contain no modalities. What is particularly useful, is the case where all formulas are of the form $s = t$. For this case, we add the following sound congruence rule:

```
left_atom_cong  "[| P, $H |= $F;
                   P == P';  |- P' ==> ($H |- $F) == ($H' |- $F') |]
                ==> (P, $H |- $F) == (P', $H' |- $F')"
```

where the judgment `|=` makes sure that all formulas have the form $s = t$. The judgment is defined in terms of Horn clauses similarly to the way the S4 sideconditions above were handled:

```
AseqE   "|="
AseqL   "$G |= $H ==> s=t,$G |= $H"
AseqR   "$G |= $H ==> $G |= s=t,$H"
```

The `left_atom_cong` rule is only added to the simplifier when used with a special simplification tactic `atom_simp_tac` which first strips the sequent of all formulas not of the form $s = t$, then simplifies the sequent making sure to prove the judgments `|=` (by means of the *subgoaler*) along the way.

In the case of the reflection rules, there are no surprises concerning their definitions:

```
eq_reflection  "|- x=y ==> (x==y)"
iff_reflection "|- P<->Q ==> (P==Q)"
```

Finally, simplification for Abelian groups is handled in essentially the same way as for the LND system.

### 8.2.3 The SIL Reasoner

The classical reasoner can not handle logics formulated in sequent calculus style (cf. Section 7.3). Isabelle is distributed with a much less developed reasoner for use with sequent calculus logics. Unfortunately, this reasoner is not flexible enough to be able to handle the extensions necessary for modal logics, in particular automatically taking care of the sideconditions formulated as Horn clauses.

We therefore write our own reasoner specifically coined at the sequent calculus encoding of SIL. The basic "look and feel" of the classical reasoner is tried duplicated, e.g., by making it possible to remove/add rules to the reasoner in a uniform way. Furthermore, it is written so as to take care of the above mentioned sideconditions, as well as the sideconditions concerning ri/cf, in a transparent way.

Finally, as in the case of the classical reasoner for the LND system, when new (defined) modalities are introduced, L-/R-rules for these are added (if possible). This is in particular interesting if they satisfy the chop-subformula as discussed in Section 5.1.1.

## 8.3 Related Work

We will in this section discuss related work on encodings of modal and interval logics in Isabelle — as well as other theorem provers.

### 8.3.1 Modal Logic

In Chapter 3 we discussed how it in general was inherently difficult to define "nice" proof systems for modal logics (with the notable exception of Hilbert systems). This difficulty is of course no less present when trying to encode a modal logic system in a theorem prover.

An attempt which considers both theoretical and practical aspects is [AHMP92], where a natural deduction encoding of S4 is considered. Formulas are split in two syntactic categories which makes a sound encoding possible. Unfortunately, this approach is not easily generalized, hence it is not clear how to encode, e.g., S5.

Parts of our sequent calculus encoding of SIL were inspired by the undocumented modal logic encodings distributed with Isabelle (for K, T, S4 and S5). These encodings are reasonable despite the somewhat cumbersome handling of the sideconditions.

Finally, the LND encoding of SIL was inspired by the LND encodings of some of the simple modal logics (K, T, S4 and S5) as discussed in [BMV97, BMV98a, Vig00]. A prominent feature of these encodings is their modularity, which means that one logic can be defined as a conservative extension of another — corresponding to the definition of the simple modal logics in Hilbert systems.

### 8.3.2 Interval Logic

The first substantial attempt at encoding ITL and DC in a theorem prover was that of Skakkebæk [SS94, Ska94a]. There, a semantic encoding is carried out in PVS [OSR93], giving PC/DC.

As the encoding is semantic, reasoning is done directly in the higher-order meta-logic of PVS. This essentially means that not much work on a logical presentation for ITL/DC is necessary — much of the effort of [Ska94a] is on developing a convenient "front-end" to PVS giving the illusion of reasoning directly in ITL/DC. This is not always successful and the user must therefore have a considerable knowledge of the meta-logic to prove given formulas. These aspects are not present in syntactic encodings such as those discussed in this chapter. On the other hand, the semantic encoding gives a great advantage in connection with the use of decision procedures (which are an integral part of PVS). This is, e.g., the case for the use of a decision procedure for arithmetic (SVC) but Skakkebæk also implements a decision procedure for a simple subset of DC. This decidable subset of DC was found in [ZHS93] where a number of undecidability results were discussed as well. The latter results indicate that it is not likely that more complicated decidable subsets of DC exist.

Two small case-studies are conducted in PC/DC in [Ska94a]; one concerning a simple gas burner, another concerning a railway crossing. A case-study concerning properties of a steam boiler is carried out in PC/DC in [Hei99]. Extending PC/DC

with a theory for traces, a proof of Fischer's mutual exclusion protocol is carried out in [PH97].

In [MXW96], NL and MVC [ZL94] are semantically encoded in PVS. This work is inspired by, and has a lot in common with, the work of Skakkebæk.

Finally, on a somewhat different note, we want to mention the system Tempura [Mos86]. Tempura is not a theorem prover but a programming language based on a subset of discrete ITL. In [Mos86] a interpreter for Tempura is discussed.

### Isabelle

The work most closely related to what have been discussed in this chapter is that of Heilmann [Hei99]. Heilmann considers an encoding of ITL and DC in Isabelle (Isabelle/DC) using a sequent calculus system. (He does not consider LND systems.)

His encoding has many similarities with our sequent calculus encoding of SIL (as discussed in Section 8.2) but also some notable differences. These differences are mainly due to variations in the theoretical foundations. Heilmann bases his encoding on the FOL sequent calculus encoding of Isabelle, on top of which he simply adds the Hilbert axioms and rules for ITL and DC "as is". In other words, he does not include any proof theoretical considerations concerning interval logic as a basis for his encoding.

Heilmann does include a modified reasoner for his sequent calculus as well as making additions to the simplifier. The automation achieved hereby makes up for some of the drawbacks of what is (in many respects) a modified Hilbert presentation for ITL/DC.

An interesting part of Heilmann's encoding, though, is the incorporation of the SVC decision procedure for arithmetic as well as the decision procedure for a subset of DC (as discussed above in connection with PVS). Unfortunately, Isabelle is not as well-suited for this as PVS.

The steam boiler case-study mentioned above is also conducted in the Isabelle encoding of [Hei99]. This requires the introduction of many new types and constructs which unfortunately thwarts the Isabelle encoding considerably. (An approach which, perhaps, would yield a nicer encoding is discussed in Section 9.7.)

CHAPTER 9

# Applications

In this chapter we discuss how properties of a number of examples (small case-studies) have been formulated and proved using the Isabelle encodings of the previous chapter. The examples are discussed/mentioned in [Ras01c].

Our goal is to "test" the encodings: How (in)convenient are they to reason in? Which is best? What could be done better? And how?

We start in Section 9.1 by briefly discussing necessary initial developments for the use of the encodings. Then, in Sections 9.2 through 9.5, we consider a number of examples concerning properties of a "converse" modality, an oscillator, a gas burner and the deadline driven scheduler. In Section 9.6 we discuss the lessons learned and identify strengths and weaknesses. We then in Section 9.7 sketch how the use of Isabelle/HOL would increase the scalability of the approach. Finally, in Section 9.8 we conclude and give directions for future work.

## 9.1 Initial Developments

The previous chapter gave an overview of the definitions, constructs, etc., necessary for encoding SIL in both a sequent calculus system and a LND system in Isabelle. Furthermore, in case of the LND system, an overview of encoding SDC, ITL, DC and NL was given as well.

For actually using these encodings, many basic results (i.e., theorems and derived rules) must be established. We already hinted at this in the previous chapter, when, e.g., in connection with setting up the simplifier, we mentioned how many results should be proved and added to the simplifier.

We also need derived rules for the basic connectives as well as the abbreviated modalities. In the case of the sequent calculus encoding we, e.g., derive L- and R-rules for $\wedge$ as follows (note that more general rules with no $\ell$ constraints are unsound):

```
Goal "[| RI x; $D, P^(Q & len=x), R^(Q & len=x), $F |- $E |]
          ==> $D, (P&R)^(Q & len=x),            $F |- $E";

Goal "[| RI x; $E |- $G, P^(Q & len=x),     $F;
               $E |- $G, R^(Q & len=x),     $F |]
          ==> $E |- $G, (P&R)^(Q & len=x), $F";
```

It should be noted that these rules satisfy the chop-subformula property discussed in Section 5.1.1, thus making them suitable for use in the SIL reasoner. Many other rules concerning the interplay between $\frown$ and the Boolean operators as well as $\ell$ are also derived.

In the case of the LND encoding we have already mentioned how we derive I- and E-rules for abbreviated modalities such as $\lceil\frown\rceil$. Other useful abbreviated modalities are:

```
ssub          :: "o => o"             ("<S>_" [50] 50)
asub          :: "o => o"             ("[S]_" [50] 50)
ssub_def      "<S>P == True|^|P|^|True"
asub_def      "[S]P == ~(<S>(~P))"
```

These two modalities are used to express properties concerning some ( `<S>`) and all (`[S]`) subintervals of an interval. I-/E-rules are derived and in the case of, e.g., `<S>` we get:

```
Goal "[| <i,k>:fwd; <k,l>:P; <k,l>:fwd; <l,j>:fwd |] ==> <i,j>:<S>P";

Goal "[| <i,j>:<S>P; <i,j>:fwd; !!k l. [| <i,k>:fwd; <k,l>:P;
         <k,l>:fwd; <l,j>:fwd |] ==> <m,n>:R |] ==> <m,n>:R";
```

We also need very basic theorems concerning $\ell$ such as the following which "separates" the $\ell$ and the ordering $\leq$:

```
Goal "RI s ==> <i,j>:s<=len <-> (EX x. len=x & s<=x)";
```

Utilizing this theorem we can now prove useful rules such as:

```
Goal "[| RI s; RI t; <i,k>:s<=len; <k,j>:t<=len |] ==> <i,j>:s+t<=len";
```

To prove such a theorem we have to manipulate the formulas concerning `len` until we end up with a purely arithmetic property, which in this case is

$$x \leq y \wedge z \leq u \rightarrow x + z \leq y + u.$$

When working in the abstract domain setting of totally ordered infinite fields, we have to proof this property by hand as well. This can be tedious at times. We will return to this point in Section 9.6.

Although the above example is formulated in the LND system, similar theorems must be proved in the sequent calculus system too.

## 9.2 The Converse Modality

In this section we will consider a particular example in greater detail. The example is useful for illustrating some important differences between reasoning in the sequent calculus system and in the LND system.

In Section 4.1.5, the connections between SIL and arrow logic/relation algebra were discussed. The results relied on the capability to define an abbreviated unary modality $^{-1}$ (read: converse) in SIL which "reverses" the direction of an interval:

$$\phi^{-1} \quad \widehat{=} \quad (\exists x)(\ (\ell = x) \wedge (\ (\ell = 0) \wedge (\ell = x)^\frown \phi\ )^\frown \text{true}\ ),$$

where $x$ is some variable not free in $\phi$. As mentioned in Section 4.1.5, it is straightforward to semantically show that $\phi^{-1}$ is satisfied on an interval $(i, j)$ iff $\phi$ is satisfied on the interval $(j, i)$.

In this section we will consider proving some simple properties of $^{-1}$:

1) $(\phi^{-1})^{-1} \leftrightarrow \phi$,

2) $(\phi^\frown \psi)^{-1} \leftrightarrow (\psi^{-1 \frown} \phi^{-1})$.

The fact that these properties hold are not surprising considering the relationship to arrow logic and relational algebra.

### 9.2.1 Labelled Natural Deduction

In this section we consider proving 1) and 2) in the LND system for SIL.

A crucial observation is that we can derive I-/E-rules for $^{-1}$:

$$\frac{(i, j) : \phi}{(j, i) : \phi^{-1}}\ {}^{-1}I \qquad\qquad \frac{(i, j) : \phi^{-1}}{(j, i) : \phi}\ {}^{-1}E\ .$$

Utilizing these rules, the proofs of 1) and 2) become much simpler. First, we will discuss the proof of $^{-1}I$ in some detail.

#### "Pen and Paper" Proof

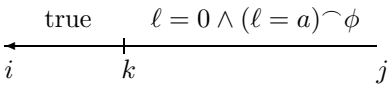
We start by giving an informal "pen and paper" proof of $^{-1}I$.

We want to show $(j, i) : \phi^{-1}$ (read: "$\phi^{-1}$ holds on the signed interval $(j, i)$") under the assumption $(i, j) : \phi$. We divide the proof in three parts.

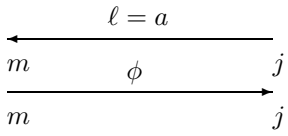
**A.** First, we notice that any interval has a length; assume the length of $(j,i)$ is $a$, i.e., $(j,i) : \ell = a$. Thus, after expanding the definition of $^{-1}$ and instantiating the existential quantifier, we are left with proving

$$(j,i) : ((\ell = 0) \wedge (\ell = a)^\frown\phi)^\frown\text{true}.$$

This can be illustrated as follows:




**B.** Hence, we have to find a $k$ such that $(k,i) : \text{true}$ and $(j,k) : (\ell = 0) \wedge (\ell = a)^\frown\phi$. Since true holds on any interval and $\ell = 0$ only holds on point intervals we take $k$ to be $j$ and thus have to prove that $(j,j) : (\ell = a)^\frown\phi$, viz.




**C.** Hence, we now have to find an $m$ such that $(j,m) : \ell = a$ and $(m,j) : \phi$. But $(j,i) : \ell = a$ and we are thus done as $(i,j) : \phi$ by assumption.

### Isabelle Proof

We will now consider a proof of $^{-1}I$ in Isabelle/LSIL.

We define $^{-1}$ straightforwardly in Isabelle as follows:

```
conv          :: o => o
conv_def      "conv(P) == (EX x. (len=x) & (len=0 & (len=x)^P)^True)"
```

Thus, $\phi^{-1}$ is written `conv(P)`.

First, we need two simple lemmas (very easily derivable),

```
len_ex  "(<i,j>:EX x. len=x ==> <i,j>:P) ==> <i,j>:P";

TrueI   "<i,j>:True";
```

saying, respectively, that any interval has some length and that true holds on any interval.

The following proof of $^{-1}I$ is taken verbatim[1] from an Isabelle session. We start by stating the goal:

[1] With the omission of some diagnostic output and minor pretty-printing.

```
> Goalw [conv_def] "<i,j>:P ==> <j,i>:conv(P)";
<i,j> : P ==> <j,i> : conv(P)
 1. <i,j> : P ==> <j,i> : EX x. len = x & (len = 0 & len = x ^ P) ^ True
```

Note how the definition of `conv` is expanded as indicated in the statement of the goal (via `[conv_def]`). We now successively refine subgoals by applying resolution tactics using suitable rules, possibly solving subgoals by assumption. We divide the proof in three parts corresponding to the "pen and paper" proof.

**A.**

```
> by (resolve_tac [len_ex] 1);
 1. [| <i,j> : P; <j,i> : EX x. len = x |]
    ==> <j,i> : EX x. len = x & (len = 0 & len = x ^ P) ^ True

> by (eresolve_tac [exE] 1);
 1. !!x. [| <i,j> : P; RI x; <j,i> : len = x |]
         ==> <j,i> : EX x. len = x & (len = 0 & len = x ^ P) ^ True

> by (eresolve_tac [exIRI] 1);
 1. !!x. [| <i,j> : P; <j,i> : len = x |]
         ==> <j,i> : len = x & (len = 0 & len = x ^ P) ^ True

> by (resolve_tac [conjI] 1);
 1. !!x. [| <i,j> : P; <j,i> : len = x |] ==> <j,i> : len = x
 2. !!x. [| <i,j> : P; <j,i> : len = x |]
         ==> <j,i> : (len = 0 & len = x ^ P) ^ True
```

**B.**

```
> by (assume_tac 1);
 1. !!x. [| <i,j> : P; <j,i> : len = x |]
         ==> <j,i> : (len = 0 & len = x ^ P) ^ True

> by (resolve_tac [chopI] 1);
 1. !!x. [| <i,j> : P; <j,i> : len = x |]
         ==> <j,?k4(x)> : len = 0 & len = x ^ P
 2. !!x. [| <i,j> : P; <j,i> : len = x |] ==> <?k4(x),i> : True

> by (resolve_tac [TrueI] 2);
 1. !!x. [| <i,j> : P; <j,i> : len = x |]
         ==> <j,?k4(x)> : len = 0 & len = x ^ P

> by (resolve_tac [conjI] 1);
 1. !!x. [| <i,j> : P; <j,i> : len = x |] ==> <j,?k4(x)> : len = 0
 2. !!x. [| <i,j> : P; <j,i> : len = x |] ==> <j,?k4(x)> : len = x ^ P
```

**C.**

```
> by (resolve_tac [zero] 1);
 1. !!x. [| <i,j> : P; <j,i> : len = x |] ==> <j,j> : len = x ^ P

> by (resolve_tac [chopI] 1);
 1. !!x. [| <i,j> : P; <j,i> : len = x |] ==> <j,?k8(x)> : len = x
 2. !!x. [| <i,j> : P; <j,i> : len = x |] ==> <?k8(x),j> : P

> by (assume_tac 1);
 1. !!x. [| <i,j> : P; <j,i> : len = x |] ==> <i,j> : P

> by (assume_tac 1);
No subgoals!
```

Notice that the two schematic variables `?k4(x)` and `?k8(x)` correspond, respectively, to $k$ and $m$ in the "pen and paper" proof.

It should be clear from the above example that the proofs in Isabelle/LSIL are very close to the abstraction level of "pen and paper" reasoning.

### Automation

In this section we prove 1) and 2) in Isabelle/LSIL, taking advantage of some of the search tactics of the classical reasoner. The I- and E-rules for $^{-1}$, and subsequently 1) and 2), can be proven in one line each using `fast_tac`/`slow_tac`:

```
Goalw [conv_def] "<i,j>:P ==> <j,i>:conv(P)";
by (resolve_tac [len_ex] 1);
by (fast_tac (claset() addEs [exE] addIs [exIRI]) 1);
qed "convI";

Goalw [conv_def] "<i,j>:conv(P) ==> <j,i>:P";
by (slow_tac (claset() addEs [exE]@uniqs@zeros) 1);
qed "convE";

Goal "<i,j>:conv(conv(P)) <-> P";
by (fast_tac (claset() addIs [convI] addDs [convE]) 1);
qed "conv_conv";

Goal "<i,j>:conv(P^Q) <-> conv(Q)^conv(P)";
by (fast_tac (claset() addIs [convI] addDs [convE]) 1);
qed "conv_chop";
```

A little explanation is required: `uniqs` contains the two rules $S1$ and $S2$ (cf. Sections 6.1.2 and 8.1.1) whereas `zeros` contains rules derived from $S1/S2$ in the cases where $s = 0$. Hence, one of the rules of `zeros` is:

```
Goal "[| <i,k>:len=0; <k,j>:P |] ==> <i,j>:P";
```

The search tactic `fast_tac` performs a depth-first search utilizing the rules of `claset()` plus the additional I-/E-rules added explicitly. The tactic `slow_tac` is a variation of `fast_tac` which does backtracking over proof by assumption. This is sometimes necessary as there might be more possibilities in the list of assumptions if there are many schematic variables.

The use of the classical reasoner in the above script nicely illustrates the principle of adding derived rules for abbreviated modalities, such that reasoning takes place on a higher level of abstraction.

### 9.2.2 Sequent Calculus

Proving 1) and 2) in the sequent calculus encoding is considerably harder than in the LND encoding. The most disturbing part is the lack of a simple connection between the parts of the proof and how a semantic argument would go, cf. the pen and paper proof above. In particular, it is not possible to refer to the intervals $(i, j)$ in the logic, hence we cannot derive I-/E-rules for $^{-1}$.

As a consequence, we basically have to work with the full formula with the $^{-1}$ definition expanded. Despite the simple-looking appearance of 1) and 2), if expanded, e.g., 2) reads as follows:

$$\begin{array}{l}(\exists x)((\ell = x) \wedge ((\ell = 0) \wedge (\ell = x)^\frown(\phi^\frown\psi))^\frown\mathrm{true}) \quad \leftrightarrow \\ \quad ((\exists x)((\ell = x) \wedge ((\ell = 0) \wedge (\ell = x)^\frown\psi)^\frown\mathrm{true}))^\frown \\ \qquad ((\exists x)((\ell = x) \wedge ((\ell = 0) \wedge (\ell = x)^\frown\phi)^\frown\mathrm{true})).\end{array}$$

This means that we cannot reason independently of subintervals but have to "collapse" them (by means of the axiom $\ell = s + t \leftrightarrow (\ell = s)^\frown(\ell = t)$ and related techniques). Furthermore, it means that proofs get more complex as it is more difficult to modularize proofs and separate concerns.

Assuming (and not including) the initial development sketched in Section 9.1, the proof of 1) and 2) took up a total of approximately 200 lines in the proof script. This is despite the fact that both the SIL Reasoner and the simplifier were utilized. Furthermore, not only is the proof longer, it is also less intuitive and took some ingenuity to complete.

## 9.3 Oscillator

In this section we consider a small example concerning a liveness property (i.e., something not expressible in ITL) of a simple oscillator. The example is discussed in [Ras99b] where a (detailed) pen and paper proof is carried out in the Hilbert proof system for SIL.

To be more concrete, the example concerns a very simple oscillator with output $Z$. We want to specify that $Z$ oscillates between 0 and 1 forever. We can specify this abstract liveness property in the following manner:

$$\mathit{Spec} \quad \widehat{=} \quad \boxdot(\diamondsuit\lceil Z\rceil \wedge \diamondsuit\lceil\neg Z\rceil),$$

where $\boxminus$ and $\diamondminus$ are two abbreviated modalities defined such that $\boxminus\phi$ holds on the current interval iff $\phi$ holds on all future intervals and $\diamondminus\phi$ holds iff $\phi$ holds on some future interval. For a discussion on how these modalities are defined and the reason behind, see [Ras99a, Ras99b]. We will here not go into the details, the intuition given should be sufficient.

We now want to describe an implementation which satisfies the specification. A possible implementation is one where we make sure that $Z$ alternates within a specified positive delay $\delta$:

$$Impl \quad \widehat{=} \quad \boxminus(\lceil Z \rceil \rightarrow \ell \leq \delta) \wedge \boxminus(\lceil \neg Z \rceil \rightarrow \ell \leq \delta).$$

To formally prove the correctness of the implementation with respect to the specification is to show that *Impl* implies *Spec*.

Below we give the Isabelle syntax of $\boxminus$ (`[F]`) and $\diamondminus$ (`<F>`):

```
sfut          :: "o => o"                    ("<F>_" [50] 50)
afut          :: "o => o"                    ("[F]_" [50] 50)
```

Following the usual recipe we derive I- and E-rules for both modalities.

The specification and implementation can now be formulated as follows in Isabelle/LSIL:

```
spec    :: "s => o"            ("(SPEC _)")
impl    :: "[s,D] => o"        ("(IMPL _ _)")

spec_def  "SPEC Z   == [F](<F>‘Z‘ & <F>‘NOT Z‘)"
impl_def  "IMPL Z d == [F](‘Z‘ --> len<=d) & [F](‘NOT Z‘ --> len<=d)"
```

The proof obligation is thus:

```
Goal "RI d ==> <i,j>:0<d --> ((IMPL Z d) --> (SPEC Z))";
```

The proof in Isabelle/LSIL utilizes the I- and E- rules for the “future modalities” to break up the proof in manageable peaces, taking advantage of the classical reasoner several places. A number of basic theorems concerning arithmetic, the $\ell$ constant and duration terms have to be proved from scratch. Most of these are of a general nature, though, which make them likely to be useful in other developments.

Finally, this example is formulated in “pure” SIL, i.e., both forward and backward intervals are taken into account. In other words, the modality `<F>` (`[F]`) talks about some (all) *signed* interval(s). But as the directions of the intervals have no inherent meaning in the example, this only complicates the proof. It would have been simpler to formulate the example using, e.g., NL. To conclude, it seems that only if the directions of intervals have an inherent meaning, should signed intervals be utilized in the specification of a problem.

## 9.4 Gas Burner

In this section we will consider an example which is based on the ITL encoding in Isabelle/LSIL as discussed in Section 8.1.4. The example is the classical Gas Burner example; the motivating example for the introduction of Duration Calculus [ZHR91].

Consider a gas burner which has to satisfy the following safety requirement: Gas only leaks ($L$) from the burner for a twentieth of the time when monitored for at least one minute, viz.

$$\ell \geq 60 \rightarrow 20 \int L \leq \ell.$$

Suppose that a design of the control system for the gas burner assures that 1) gas is turned off at most one second after gas has started leaking,

$$\Box(\lceil L \rceil \rightarrow \ell \leq 1),$$

and 2) gas is not turned on until 30 seconds have elapsed since the gas was leaking,

$$\Box((\Diamond\lceil L \rceil)^\frown(\Diamond\lceil \neg L \rceil)^\frown(\Diamond\lceil L \rceil) \rightarrow \ell \geq 30).$$

(Here, $\Box$, $\Diamond$ and $^\frown$ are the ITL versions of the modalities.)

In Isabelle/LSIL, these criteria can be formulated as follows:

```
safe     :: "s => o"            ("(Safe _)")
des1     :: "s => o"            ("(Des1 _)")
des2     :: "s => o"            ("(Des2 _)")

safe_def "Safe L == 20*dur(L)<=len"
des1_def "Des1 L == [S](`L` --> len<=1)"
des2_def "Des2 L == [S]((<S>`L`)|^|(<S>`NOT L`)|^|(<S>`L`)
                                                --> 30<=len)"
```

The proof obligation is thus:

```
Goal "[| <i,j>:fwd; <i,j>:Des1 L; <i,j>:Des2 L; <i,j>:60<=len |]
     ==> <i,j>:Safe L";
```

The proof idea is here similar to that of the oscillator — this time working with the "subinterval" modalities instead of the "future" modalities. We still have to prove a number of basic theorems concerning arithmetic, etc.

What have not been discussed, is what the "20", "30" and "60" in the above Isabelle encoding actually means. We are basing our development on a totally ordered infinite field, in which only the numerals 0 and 1 are defined. We therefore have to define a concrete syntax for additionally numerals:

```
syntax
  two    :: D            ("2")
  three  :: D            ("3")
  ...
  ten    :: D            ("10")

  twenty :: D            ("20")
  thirty :: D            ("30")
  sixty  :: D            ("60")

translations
  "2"  ==  "1+1"
  "3"  ==  "2+1"
  ...
  "10" ==  "9+1"

  "20" ==  "2*10"
  "30" ==  "3*10"
  "60" ==  "6*10"
```

This solution is quite ad hoc and clearly not feasible in general — reasoning concerning the numerals become very tedious. If we want to be able to use such numerals in specifications we have to develop a theory for this. See Section 9.7.

## 9.5 Deadline Driven Scheduler

The Deadline Driven Scheduler (DDS) was proposed in [LL73]. The scheduler administrates a finite number of processes sharing a single processor. Each process periodically requests a constant amount of processor time (the run time) and has the period as a deadline for completion.

The DDS dynamically assigns priorities to the processes such that a process will be assigned the highest priority if its deadline for completion is the nearest. At any given time, the process with the highest priority, which has not yet fulfilled its run time, is chosen to run. Note that different processes can have different periods such that their priorities associated with their current deadlines vary dynamically with different intervals.

A necessary and sufficient condition for the DDS to be feasible is [LL73]:

$$\sum \frac{C_i}{T_i} \leq 1,$$

where $C_i$ and $T_i$ is the run time and period time for process $i$, respectively, and $0 < C_i < T_i$.

The necessity of this condition is clear whereas the sufficiency is far from obvious; [LL73] only gives an informal argument.

In [ZZ94] the DDS is formalized in DC and a formal proof of correctness is considered. A state variable $R_i$ is introduced for each process, such that $R_i$ says whether or not process $i$ is running at any given time. For a given interval of observation the criteria for correctness can now be specified in DC as follows:

$$\int R_i \ \geq \ \left\lfloor \frac{\ell}{T_i} \right\rfloor \cdot C_i.$$

The proof in [ZZ94] is by hand and many details are left out. It is thus not clear that all deductions are sound. It would be interesting to perform a rigorous formalization and prove it correct using Isabelle/LSIL. The full formalization would be quite complicated and the proof would be a non-trivial undertaking.

Here we just consider a simple property which each process must satisfy and which can be stated independently for each process: The request of a process for run time must make sure that the process does not run more than necessary in each period. For this we assume that for each process there is a state variable $S_i$ which is used to indicate whether, for any given time, the process has a standing request for more run time.

First, we assume that as long as a process is running it has a standing request for more run time:

$$\Box(\lceil R_i \rceil \rightarrow \lceil S_i \rceil).$$

Now, assume that the end point of the current interval is the beginning point of a new period for the process. The following says that if the total run time of the process in the new period is $C_i$, then it should not have a standing request for more run time for the remainder of the period:

$$\neg\Diamond_r(\ell \leq T_i \wedge (\textstyle\int R_i = C_i) \mathbin{\frown\!\!\mid} \lceil S_i \rceil).$$

What we want to show is thus:

$$\Box_r(\ell \leq T_i \rightarrow \textstyle\int R_i \leq C_i),$$

where $\Box_r$ is defined by $\Box_r\phi \ \widehat{=}\ \neg(\Diamond_r(\neg\phi))$ to mean "all right neighbourhood intervals".

In Isabelle/LSIL, the final theorem to be proved is thus the following:

```
Goal "[| <i,j>:fwd; <i,j>:0<C; RI C; <i,j>:0<T; RI T;
         <i,j>:[A](`R` --> `S`);
         <i,j>:~<R>(len<=t & (dur(R)=c)|^|`S`) |]
     ==> <i,j>:[R](len<=t --> dur(R)<=c)";
```

Again, we break the proof up according to the modalities utilizing their I-/E-rules and the classical reasoner. We have to prove some auxiliary basic theorems concerning duration terms, etc., but many of the theorems proved in connection with the previous examples can be reused conveniently.

## 9.6 Discussion

We will now further discuss the examples considered in this chapter.

The example concerning proving properties 1) and 2) of the converse modality illustrates nicely the difference between reasoning in the sequent calculus encoding for SIL and in Isabelle/LSIL. It should come as no surprise to the reader which of the encodings is the "winner": Isabelle/LSIL.

The length of the proof script in Isabelle/LSIL is much shorther, actually an order of magnitude shorter, than that of the sequent calculus encoding, and was furthermore much more intuitive to develop.

The three main reasons for choosing Isabelle/LSIL over the sequent calculus encoding are:

- Reasoning in Isabelle/LSIL is much more intuitive; the intervals, which are part of the logic, can easily be visualized and the connection to the semantics is much clearer.
- A higher degree of automation is possible in Isabelle/LSIL; this fact owes a lot to the proper natural deduction system defined for SIL.
- Isabelle is inherently a system for doing reasoning in natural deduction systems; the sequent calculus encoding can seem less natural to use.

Of course, the converse modality example is very small and not a typical specification/verification problem. Thus, it would be premature to conclude that everything is proven so easily in Isabelle/LSIL. We will get back to this point shortly.

For the remainder of this discussion, it will be convenient to introduce the notions of *formula-level* and *term-level* reasoning. Formula-level reasoning is concerned with formulas and their relationship expressed by operators and modalities whereas term-level reasoning is concerned with terms and their relationships expressed by $=$, $\leq$ and $<$.

The converse modality example is almost solely a formula-level problem, in particular is the reasoning with $\frown$ central. This puts the ease of reasoning in Isabelle/LSIL in an additional perspective: The labels (the intervals) of the LND system are crucial for the simple reasoning with $\frown$ as they can be referred to directly — contrary to the sequent calculus system.

On the other hand, term-level reasoning almost solely takes place on the same interval, hence the possibility to refer to the intervals is not very important. In other words, in this case we do not gain much help from the LND system. Furthermore, much term-level reasoning is not even related to interval logic as such (e.g., pure arithemtic reasoning).

The distinction between formula- and term-level reasoning is useful when discussing the three examples: The Oscillator, The Gas Burner and the simple property of the DDS. All three examples require amounts of both formula- and term-level reasoning. In the case of formula-level reasoning, the advantages of the LND system are obvious. In the case of term-level reasoning we had to go through the same tedious

reasoning as we would have in the case of the sequent calculus system. Even if a lot of automation is not possible in parts of the proof it is still intuitively easier to reason as the proofs are closer to informal "pen and paper" reasoning, though.

In conclusion, formula-level reasoning is greatly improved by the LND formalism whereas term-level reasoning still can be quite tedious.

What can be done to improve term-level reasoning? In Section 9.1 we mentioned how many properties concerning $\ell$ required manipulation involving $\ell$ followed by purely arithmetic reasoning. The same is the case for duration terms $\int S$. The reasoning concerning the interval dependent terms ( $\ell$ and $\int S$) seems unavoidable whereas arithmetic reasoning (including representation of integer numerals) has great room for improvement. A solution will be discussed in the following section.

## 9.7 Isabelle/HOL

Isabelle/HOL [NPW01] is an encoding of Higher-Order Logic in Isabelle. The encoding can be thought of as lifting the meta-logic of Isabelle to the object-logic level. Isabelle/HOL has many similarities with the HOL system [GM93].

Isabelle/HOL is the most well-developed theory of Isabelle. Within it, theories for natural numbers, integers, reals, sets, maps, relations, functions, datatypes and much more has been developed. This makes Isabelle/HOL an ideal choice for larger specification/verification tasks as many of the concepts taken for granted in informal specifications, in many cases exist in Isabelle/HOL already.

We will in this section discuss how we can utilize Isabelle/HOL in the context of interval logic. More specifically, we will consider how we can "port" Isabelle/LSIL to Isabelle/HOL.

Basically, there are no essential difficulties in doing this. The main work to be done is concerned with fitting the self-contained theory Isabelle/LSIL in the more complex system Isabelle/HOL. The obstacles include such trivial things as handling name-clashes, etc.

Below we give the core part of the theory file used for extending Isabelle/HOL with a LND system for SIL:

```
LSILHOL = Real +

types
  T

consts
  LF      :: "[T, T, bool] => prop"        ("(<_,_> : (_))" [6,6,5] 4)
  RI      ::      'a::term => prop         ("(RI _)")
  CF      ::          bool => prop         ("(CF _)")

  chop    :: "[bool, bool] => bool"        (infixr "^" 38)
  len     :: real
```

We only define one new type here, namely $T$. The types for FOL already exist in Isabelle/HOL; in particular, the type $o$ for formulas is in Isabelle/HOL named *bool*. Furthermore, all FOL operators and quantifiers are already defined as part of Isabelle/HOL. Hence, we only define the chop modality as well as the `len` constant.

The `Real` theory of Isabelle/HOL (on which the above theory is based) includes a development of real arithmetics. Thus, we have defined $\ell$ as having the type *Real* (we could also have chosen the natural numbers or the integers). We can then take advantage of a substantial amount of results on arithmetics. Furthermore, a concrete syntax is defined for (integer) numerals, such that, e.g., 3948234 is written `#3948234`. Such numerals can be included as integral parts in arithmetic expressions.

As we are working in a higher-order logic, $=$ is defined for all (logical) types, including *bool*. This means that we can do without `<->` altogether in Isabelle/HOL. This approach is a little problematic in connection with SIL, though, because of the sideconditions (concerning rigidity/chop-freeness) on the substitution rules (as these should be applicable for $=$ on Booleans as well).

We therefore introduce `<->` the "classical" FOL way:

```
"<->"     :: "[bool, bool] => bool"        (infixr 25)
iff_def   "P<->Q   == (P-->Q) & (Q-->P)"
```

Now, all axioms and rules of the labelled system can be included as discussed in Section 8.1.1.

To take advantage of the many theories defined, and theorems proved, in Isabelle/HOL, we introduce the following meta-axiom which we can use for "importing" theorems into the labelled system:

```
rigidprop  "(RI P) ==> (<i,j>:P) == Trueprop(P)"
```

Remembering that the validity of a rigid formula is independent of the intervals, this is clearly sound.

We will now in a little more detail explain how the development of real arithmetic can help term-level reasoning:

- The theorems we have proven concerning basic arithmetic in the totally ordered infinite field already exist in the theory of the reals — plus many more.
- The simplifier in Isabelle/HOL is more powerful for reasoning on arithmetic.
- Furthermore, Isabelle/HOL includes a tactic (`arith_tac`) which calls a simple decision procedure for linear arithmetic. This is not as powerful as SVC but is programmed within Isabelle, hence soundness is guaranteed.

As an example, consider proving the following rule (which we briefly considered in Section 9.1):

```
Goal "[| RI s; RI t; <i,k>:s<=len; <k,j>:t<=len |] ==> <i,j>:s+t<=len";
```

After a little manipulation concerning $\ell$, we are left with having to prove the following simple arithmetic property:

```
1. !!x xa. [| RI s; RI t; RI x; RI xa; <i,k>:s<=x; <k,j>:t<=xa; ... |]
           ==> <i,j>:s+t<=x+xa
```

A tactic can now utilize the axiom `rigidprop` to turn the labelled formulas into formulas of type `bool`. This is possible since, e.g., `RI (s<=x)` can be shown automatically using the assumptions and the known rules concerning rigidity.

We are then left with a goal of the following form:

```
1. !!x xa. [| s<=x; t<=xa; ... |] ==> s+t<=x+xa
```

This goal can be solved automatically by `arith_tac`, which saves quite some tedious work and the frustration of having to proof such "obvious" properties by hand.

In conclusion, the two main reasons for porting Isabelle/LSIL to Isabelle/HOL are:

1. A very substantial theory development already exists within Isabelle/HOL.
2. Arithmetic reasoning is improved considerably.

Clearly, both these factors are crucial for the scalability of theorem proving for interval logic formalisms.

## 9.8 Conclusion

We have in this chapter discussed a number of small examples concerning reasoning in Isabelle/LSIL. These examples gave a good idea of the kind of reasoning encountered when using the encoding. In particular, the distinction between formula- and term-level reasoning (where the latter again can be divided between interval-dependent and pure arithmetic reasoning) is convenient.

The examples are all fairly small, so the question is, does the approach scale to larger examples and case-studies? Our claim is, yes, it does — if the port to Isabelle/HOL is carried out in full.

A good way of supporting this claim would be to mechanize the full proof of correctness of the Deadline Driven Scheduler. Such a mechanization could fruitfully be based on the proof in [ZH01] which refines and clarifies that of [ZZ94]; though still on "pen and paper" level.

The possibility of using the developments of Isabelle/HOL would be crucial for the feasibility of the undertaking. We, e.g., have to be able to reason of expressions such as (cf. Section 9.5):

$$\sum \frac{C_i}{T_i} \leq 1 \qquad \text{and} \qquad \int R_i \;\geq\; \left\lfloor \frac{\ell}{T_i} \right\rfloor \cdot C_i,$$

which requires a substantial development of arithmetic.

# Bibliography


[A+01] Tobias Amnell et al. UPPAAL — Now, Next, and Future. In *Modelling and Verification of Parallel Processes*, volume 2067 of *Lecture Notes in Computer Science*, pages 99–124. Springer-Verlag, 2001.

[AH91] Rajeev Alur and Thomas A. Henzinger. Logics and Models of Real-Time: A Survey. In *Real-Time: Theory and Practice*, volume 600 of *Lecture Notes in Computer Science*, pages 74–106. Springer-Verlag, 1991.

[AHMP92] A. Avron, F. Honsell, I. Mason, and R. Pollack. Using Typed Lambda Calculus to Implement Formal Systems on a Machine. *Journal of Automated Reasoning*, 9:309–354, 1992.

[AKN+96] H. Andréka, Á. Kurucz, I. Németi, I. Sain, and A. Simon. Causes and Remedies for Undecidability in Arrow Logics and in Multi-Modal Logics. In M. Marx, L. Pólos, and M. Masuch, editors, *Arrow Logic and Multi-Modal Logic*, CSLI Publications, pages 63–99. Cambridge University Press, 1996.

[All83] James F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[All84] James F. Allen. Towards a General Theory of Action and Time. *Artificial Intelligence*, 23:123–154, 1984.

[And86] Peter B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*. Academic Press, 1986.

[And01] Peter B. Andrews. Classical Type Theory. In Robinson and Voronkov [RV01], pages 965–1007.

[B+89] Dines Bjørner et al. A ProCoS Project Description: ESPRIT BRA 3104. *Bulletin of the EATCS, G. Rozenberg (Ed.)*, (39):60–73, 1989.

[Bar84] Henk P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. North-Holland, Revised edition, 1984.

[BDS96] Clark W. Barrett, David L. Dill, and Aaron Stump. Validity Checking for Combinations of Theories with Equality. In *Formal Methods in Computer Aided Design, FMCAD'96*, volume 1166 of *Lecture Notes in Computer Science*, pages 187–201. Springer-Verlag, 1996.

[BDS00] Clark W. Barrett, David L. Dill, and Aaron Stump. A Framework for Cooperating Decision Procedures. In *Automated Deduction, CADE-17*, volume 1831 of *Lecture Notes in Artificial Intelligence*, pages 79–98. Springer-Verlag, 2000.

[Bel82] N. Belnap. Display logic. *Journal of Philosophical Logic*, 11:375–417, 1982.

[Ben86] E. Bencivenga. Free Logics. In D.M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume III, pages 373–426. Reidel Publishing Company, 1986.

[BG01] Henk Barendregt and Herman Geuvers. Proof Assistants Using Dependent Type Systems. In Robinson and Voronkov [RV01], pages 1149–1238.

[Bla00] Patrick Blackburn. Representation, Reasoning, and Related Structures: a Hybrid Logic Manifesto. *Logic Journal of the IGPL*, 8(3):339–365, 2000.

[BM88] R.S. Boyer and J.S. Moore. *A Computational Logic Handbook*. Academic Press, 1988.

[BM92] Mario R.F. Benevides and Thomas S.E. Maibaum. A constructive presentation for the modal connective of necessity ( $\Box$). *Journal of Logic and Computation*, 2(1):31–50, 1992.

[BMV97] David Basin, Seán Matthews, and Luca Viganò. Labelled Propositional Modal Logics: Theory and Practice. *Journal of Logic and Computation*, 7(6):685–717, 1997.

[BMV98a] David Basin, Seán Matthews, and Luca Viganò. Labelled Modal Logics: Quantifiers. *Journal of Logic, Language and Information*, 7(3):237–263, 1998.

[BMV98b] David Basin, Seán Matthews, and Luca Viganò. Natural Deduction for Non-Classical Logics. *Studia Logica*, 60(1):119–160, 1998.

[BN98] Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[BP01] Giampaolo Bella and Lawrence C. Paulson. Mechanical Proofs about a Non-repudiation Protocol. In *Theorem Proving in Higher Order Logics, TPHOLs 2001*, volume 2152 of *Lecture Notes in Computer Science*, pages 91–104. Springer-Verlag, 2001.

[BS84] Robert A. Bull and Krister Segerberg. Basic Modal Logic. In Gabbay and Guenthner [GG84], pages 1–88.

[BZ97] Rana Barua and Zhou Chaochen. Neighbourhood Logics : NL and NL $^2$. Research Report 120, UNU/IIST, August 1997.

[C$^+$86] Robert L. Constable et al. *Implementing Mathematics with the Nuprl Proof Development System*. Prentice-Hall International, 1986.

[Cer93] Claudio Cerrato. Modal Sequents for Normal Modal Logics. *Mathematical Logic Quarterly*, 39:231–240, 1993.

[Che80] Brian F. Chellas. *Modal Logic: An Introduction*. Cambridge University Press, 1980.

[Coq00] The Coq Proof Assistant, 2000. `http://coq.inria.fr`.

[CS01] Edmund M. Clarke and Bernd-Holger Schlingloff. Model Checking. In Robinson and Voronkov [RV01], pages 1635–1790.

[Cur52] Haskell B. Curry. The Elimination Theorem when Modality is Present. *The Journal of Symbolic Logic*, 17:249–265, 1952.

[Dav01] Martin Davis. The Early History of Automated Deduction. In Robinson and Voronkov [RV01], pages 3–15.

[Doŝ85] Kosta Doŝen. Sequent-Systems for Modal Logic. *The Journal of Symbolic Logic*, 50(1):149–159, 1985.

[Dut95a] Bruno Dutertre. Complete Proof Systems for First Order Interval Temporal Logic. In *Logic in Computer Science, LICS'95*, pages 36–43. IEEE Computer Society Press, 1995.

[Dut95b] Bruno Dutertre. On First Order Interval Temporal Logic. Technical Report CSD-TR-94-3, Department of Computer Science, Royal Holloway, University of London, February 1995.

[ER94] Marcin Engel and Hans Rischel. Dagstuhl-Seminar Specification Problem - a Duration Calculus Solution. Unpublished Manuscript, Dept. of Computer Science, Technical University of Denmark, September 1994.

[Fit83] Melvin Fitting. *Proof Methods for Modal and Intuitionistic Logics*. Reidel Publishing Company, 1983.

[Fit90] Melvin Fitting. *First-Order Logic and Automated Theorem Proving*. Springer-Verlag, 1990.

[Fle00] Jacques D. Fleuriot. On the Mechanization of Real Analysis in Isabelle/HOL. In *Theorem Proving in Higher Order Logics, TPHOLs 2000*, volume 1869 of *Lecture Notes in Computer Science*, pages 146–162. Springer-Verlag, 2000.

[FM98] Melvin Fitting and Richard L. Mendelsohn. *First-Order Modal Logic*, volume 227 of *Synthese Library*. Kluwer Academic Publishers, 1998.

[Frä97] Martin Fränzle. *Controller Design from Temporal Logic: Undecidability Need Not Matter*. PhD thesis, Christian-Albrechts-Universität Kiel, June 1997.

[Fuc63] L. Fuchs. *Partially Ordered Algebraic Systems*. Addison-Wesley, 1963.

[Gab92] Dov M. Gabbay. Elements of Algorithmic Proof. In S. Abramsky, D.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume II, pages 307–408. Oxford University Press, 1992.

[Gab96] Dov M. Gabbay. *Labelled Deductive Systems*, volume I. Oxford University Press, 1996.

[Gal86] Jean H. Gallier. *Logic for Computer Science: Foundations of Automatic Theorem Proving*. Harper & Row Publishers, 1986.

[Gar84] James W. Garson. Quantification in Modal Logic. In Gabbay and Guenthner [GG84], pages 249–308.

[Gen35] G. Gentzen. Untersuchungen über das logische Schliessen. *Mathematische Zeitschrift*, 39:176–210,405–431, 1935.

[GG84] D.M. Gabbay and F. Guenthner, editors. *Handbook of Philosophical Logic*, volume II. Reidel Publishing Company, 1984.

[Gir95] Jean-Yves Girard. Linear logic: Its syntax and semantics. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, volume 222 of *LMSLNS*, pages 1–42. Cambridge University Press, 1995.

[GLT89] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Cambridge Tracts in Theoretical Computer Science 7. Cambridge University Press, 1989.

[GM93] M.J.C. Gordon and T.F. Melham. *Introduction to HOL: A Theorem Proving Enviroment for Higher Order Logic*. Cambridge University Press, 1993.

[GM02] D.M. Gabbay and G. Malod. Naming Worlds in Modal and Temporal Logic. *Journal of Logic, Language and Information*, 11(1):29–65, 2002.

[GO00] D.M. Gabbay and N. Olivetti. *Goal Directed Algorithmic Proof Theory*. Kluwer Academic Publishers, 2000.

[Gor92] Rajeev P. Goré. *Cut-free Sequent and Tableau Systems for Propositional Normal Modal Logics*. PhD thesis, Computer Laboratory, University of Cambridge, May 1992. Technical Report 257.

[Gor00] Mike Gordon. From LCF to HOL: A Short History. In G. Plotkin, C. Stirling, and M. Tofte, editors, *Proof, Language, and Interaction: Essays in Honour of Robin Milner*, pages 169–185. MIT Press, 2000.

[Gue98] Dimitar P. Guelev. A Calculus of Durations on Abstract Domains: Completeness and Extensions. Research Report 139, UNU/IIST, May 1998.

[Ham88] A.G. Hamilton. *Logic for Mathematicians*. Cambridge University Press, Revised edition, 1988.

[Han94] Michael R. Hansen. Model-Checking Discrete Duration Calculus. *Formal Aspects of Computing*, 6:826–845, 1994.

[Har96] John Harrison. Formalized mathematics. Technical Report 36, Turku Centre for Computer Science (TUCS), 1996.

[HC68] G.E. Hughes and M.J. Cresswell. *An Introduction to Modal Logic*. Routledge, 1968.

[HC84] G.E. Hughes and M.J. Cresswell. *A Companion to Modal Logic*. Methuen, 1984.

[HC96] G.E. Hughes and M.J. Cresswell. *A New Introduction to Modal Logic*. Routledge, 1996.

[Hei99] Søren T. Heilmann. *Proof Support for Duration Calculus*. PhD thesis, Dept. of Information Technology, Technical University of Denmark, January 1999.

[HMM83] J.Y. Halpern, B. Moszkowski, and Z. Manna. A Hardware Semantics based on Temporal Intervals. In *Automata, Languages, and Programming, ICALP'83*, volume 154 of *Lecture Notes in Computer Science*, pages 278–291. Springer-Verlag, 1983.

[HPJ00] V.F. Hendricks, S.A. Pedersen, and K.F. Jørgensen, editors. *Proof Theory. History and Philosophical Significance*, volume 292 of *Synthese Library*. Kluwer Academic Publishers, 2000.

[HS91] J.Y. Halpern and Y. Shoham. A Propositional Modal Logic of Time Intervals. *Journal of the ACM*, 38(4):935–962, October 1991.

[HY99] S. Hagihara and N. Yonezaki. Resolution Method for Modal Logic with Well-Founded Frames. In *Computer Science Logic, CSL'99*, volume 1683 of *Lecture Notes in Computer Science*, pages 277–291. Springer-Verlag, 1999.

[HZ97] Michael R. Hansen and Zhou Chaochen. Duration Calculus: Logical Foundations. *Formal Aspects of Computing*, 9(3):283–330, May–June 1997.

[Isa01] Isabelle99-2, 2001.
`http://www.cl.cam.ac.uk/Research/HVG/Isabelle/` .

[Kal01] John Arnold Kalman. *Automated Reasoning with OTTER*. Rinton Press, 2001.

[KMM00] Matt Kaufmann, Panagiotis Manolios, and J. Strother Moore. *Computer-Aided Reasoning: An Approach*. Kluwer Academic Publishers, 2000.

[KNSS95] Á. Kurucz, I. Németi, I. Sain, and A. Simon. Decidable and Undecidable Logics with a Binary Modality. *Journal of Logic, Language and Information*, 4:191–206, 1995.

[Lew90] Harry R. Lewis. A Logic of Concrete Time Intervals. In *Logic in Computer Science, LICS'90*, pages 380–389. IEEE Computer Society Press, 1990.

[LH94] Y. Lakhneche and J. Hooman. Reasoning about Durations in Metric Temporal Logic. In *Formal Techniques in Real-Time and Fault-Tolerant Systems, FTRTFT'94*, volume 863 of *Lecture Notes in Computer Science*, pages 488–510. Springer-Verlag, 1994.

[LL73] C.L. Liu and J.W. Layland. Scheduling Algorithm for Multiprogramming in a Hard Real-Time Environment. *Journal of the ACM*, 20(1):46–61, 1973.

[Mac95] Donald Mackenzie. The Automation of Proof: A Historical and Sociological Exploration. *IEEE Annals of the History of Computing*, 17(3):7–29, 1995.

[Mas92] Andrea Masini. 2-Sequent Calculus: A Proof Theory of Modalities. *Annals of Pure and Applied Logic*, 58:229–246, 1992.

[Men87] Elliott Mendelson. *Introduction to Mathematical Logic*. Wadsworth & Brooks/Cole, Third edition, 1987.

[Min90] G. Mints. Gentzen-Type Systems and Resolution Rules. In *COLOG-88*, volume 417 of *Lecture Notes in Computer Science*, pages 198–231. Springer-Verlag, 1990.

[MN90] Ursula Martin and Tobias Nipkow. Ordered Rewriting and Confluence. In *Automated Deduction, CADE-10*, volume 449 of *Lecture Notes in Artificial Intelligence*, pages 366–380. Springer-Verlag, 1990.

[Mos85] Ben Moszkowski. A Temporal Logic for Multilevel Reasoning about Hardware. *IEEE Computer*, 18(2):10–19, 1985.

[Mos86] Ben Moszkowski. *Executing Temporal Logic Programs*. Cambridge University Press, 1986.

[Mos94] Ben Moszkowski. Some Very Compositional Temporal Properties. In E.-R. Olderog, editor, *Programming Concepts, Methods, and Calculi*, volume A-56 of *IFIP Transactions*, pages 307–326. Elsevier Science, 1994.

[Mos95] Ben Moszkowski. Compositional Reasoning about Projected and Infinite Time. In *First IEEE Intl. Conf. on Engineering of Complex Computer Systems*, pages 238–245. IEEE Computer Society Press, 1995.

[MV97] Maarten Marx and Yde Venema. *Multi-Dimensional Modal Logic*, volume 4 of *Applied Logic Series*. Kluwer Academic Publishers, 1997.

[MXW96] Mao Xiaoguang, Xu Qiwen, and Wang Ji. Towards a Proof Assistant for Interval Logics. Research Report 77, UNU/IIST, October 1996.

[NGdV94] R.P. Nederpelt, J.H. Geuvers, and R.C. de Vrijer, editors. *Selected papers on Automath*, volume 133 of *Studies in Logic*. North-Holland, 1994.

[NPW01] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle's Logics: HOL*, February 2001. Part of [Isa01].

[OSR93] S. Owre, N. Shankar, and J.M. Rushby. *User Guide for the PVS Specification and Verification, Language and Proof Checker*. SRI International, February 1993.

[Pan96] Paritosh K. Pandya. Weak Chop Inverses and Liveness in Duration Calculus. In *Formal Techniques in Real-Time and Fault-Tolerant Systems, FTRTFT'96*, volume 1135 of *Lecture Notes in Computer Science*, pages 148–167. Springer-Verlag, 1996.

[Pau86] Lawrence C. Paulson. Natural Deduction as Higher-Order Resolution. *Journal of Logic Programming*, 3:237–258, 1986.

[Pau89] Lawrence C. Paulson. The Foundations of a Generic Theorem Prover. *Journal of Automated Reasoning*, 5(3):363–397, 1989.

[Pau90] Lawrence C. Paulson. Isabelle: The Next 700 Theorem Provers. In P. Odifreddi, editor, *Logic and Computer Science*, pages 361–386. Academic Press, 1990.

[Pau94] Lawrence C. Paulson. *Isabelle, A Generic Theorem Prover*, volume 828 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[Pau97] Lawrence C. Paulson. Generic Automatic Proof Tools. In Robert Veroff, editor, *Automated Reasoning and Its Applications: Essays in Honour of Larry Wos*, pages 23–47. MIT Press, 1997.

[Pau98] Lawrence C. Paulson. The Inductive Approach to Verifying Cryptographic Protocols. *Journal of Computer Security*, 6:85–128, 1998.

[Pau01a] Lawrence C. Paulson. *Isabelle's Logics*. Computer Laboratory, University of Cambridge, February 2001. Part of [Isa01].

[Pau01b] Lawrence C. Paulson. *The Isabelle Reference Manual*. Computer Laboratory, University of Cambridge, February 2001. Part of [Isa01].

[Pfe01] Frank Pfenning. Logical Frameworks. In Robinson and Voronkov [RV01], pages 1063–1147.

[PG96] Lawrence C. Paulson and Krzysztof Grabczewski. Mechanizing Set Theory. *Journal of Automated Reasoning*, 17:291–323, 1996.

[PH97] Michael Bruun Petersen and Torben Hoffmann. Mechanical Support of Real-Time Trace Logic. Master's thesis, Dept. of Information Technology, Technical University of Denmark, August 1997.

[Pla93] David A. Plaisted. Equational Reasoning and Term Rewriting Systems. In D.M. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 1, pages 273–364. Oxford University Press, 1993.

[Pnu77] Amir Pnueli. Temporal Logic of Programs. In *Logic in Computer Science, LICS'77*, pages 46–57. IEEE Computer Society Press, 1977.

[Pra65] Dag Prawitz. *Natural Deduction. A Proof-Theoretical Study*. Almquist & Wiksell, 1965.

[Ras99a] Thomas M. Rasmussen. Signed Interval Logic. In *Computer Science Logic, CSL'99*, volume 1683 of *Lecture Notes in Computer Science*, pages 157–171. Springer-Verlag, 1999.

[Ras99b] Thomas M. Rasmussen. Signed Interval Logic. Master's thesis, Dept. of Information Technology, Technical University of Denmark, January 1999.

[Ras99c] Thomas M. Rasmussen. Signed Interval Logic on Totally Ordered Domains. Unpublished note, Dept. of Information Technology, Technical University of Denmark, June 1999.

[Ras00] Thomas M. Rasmussen. Formalizing Basic Number Theory. Technical Report 502, Computer Laboratory, University of Cambridge, September 2000.

[Ras01a] Thomas M. Rasmussen. A Sequent Calculus for Signed Interval Logic. Technical Report IMM-TR-2001-06, Informatics and Mathematical Modeling, Technical University of Denmark, 2001.

[Ras01b] Thomas M. Rasmussen. An Inductive Approach to Formalizing Notions of Number Theory Proofs. In *Computer Mathematics, ASCM 2001*, volume 9 of *Lecture Notes Series on Computing*, pages 131–140. World Scientific, 2001.

[Ras01c] Thomas M. Rasmussen. Automated Proof Support for Interval Logics. In *Logic for Programming, Artificial Intelligence, and Reasoning, LPAR 2001*, volume 2250 of *Lecture Notes in Artificial Intelligence*, pages 317–326. Springer-Verlag, 2001.

[Ras01d] Thomas M. Rasmussen. Labelled Natural Deduction for Interval Logics. In *Computer Science Logic, CSL'01*, volume 2142 of *Lecture Notes in Computer Science*, pages 308–323. Springer-Verlag, 2001.

[RM+96a] Y.S. Ramakrishna, P.M. Melliar-Smith, L.E. Moser, L.K. Dillon, and G. Kutty. Interval logics and their decision procedures. Part I: An interval logic. *Theoretical Computer Science*, 166(1–2):1–47, 1996.

[RM+96b] Y.S. Ramakrishna, P.M. Melliar-Smith, L.E. Moser, L.K. Dillon, and G. Kutty. Interval logics and their decision procedures. Part II: A real-time interval logic. *Theoretical Computer Science*, 170(1–2):1–46, 1996.

[Rob65] J.A. Robinson. A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the ACM*, 12:23–41, 1965.

[RP86] Roni Rosner and Amir Pnueli. A Choppy Logic. In *Logic in Computer Science, LICS'86*, pages 306–313. IEEE Computer Society Press, 1986.

[RT99] Piotr Rudnicki and Andrzej Trybulec. On Equivalents of Well-foundedness. An Experiment in MIZAR. *Journal of Automated Reasoning*, 23:197–234, 1999.

[RV01] Alan Robinson and Andrei Voronkov, editors. *Handbook of Automated Reasoning*, volume I+II. Elsevier Science, 2001.

[RZ97] Suman Roy and Zhou Chaochen. Notes on Neighborhood Logic. Research Report 97, UNU/IIST, February 1997.

[Ser82] O.F. Serebriannikov. Gentzen's Hauptsatz for Modal Logic with Quantifiers. *Acta Philosophica Fennica*, 35:79–88, 1982.

[Sho77] Robert E. Shostak. On the SUP-INF Method for Proving Presburger Formulas. *Journal of the ACM*, 24(4):529–543, 1977.

[Sho78] Robert E. Shostak. An Algorithm for Reasoning About Equality. *Communications of the ACM*, 21(7):583–585, 1978.

[Sho79] Robert E. Shostak. A Practical Decision Procedure for Arithmetic with Function Symbols. *Journal of the ACM*, 26(2):351–360, 1979.

[Sho84] Robert E. Shostak. Deciding Combinations of Theories. *Journal of the ACM*, 31(1):1–12, 1984.

[Ska94a] Jens Ulrik Skakkebæk. *A Verification Assistant for a Real-Time Logic*. PhD thesis, Dept. of Computer Science, Technical University of Denmark, November 1994.

[Ska94b] Jens Ulrik Skakkebæk. Liveness and Fairness in Duration Calculus. In *Concurrency Theory, CONCUR'94*, volume 836 of *Lecture Notes in Computer Science*, pages 283–298. Springer-Verlag, 1994.

[Smu68] R.M. Smullyan. *First-Order Logic*. Springer-Verlag, 1968.

[SS93] G. Schmidt and T. Ströhlein. *Relations and Graphs*. Springer-Verlag, 1993.

[SS94] J.U. Skakkebæk and N. Shankar. Towards a Duration Calculus Proof Assistant in PVS. In *FTRTFT'94*, volume 863 of *Lecture Notes in Computer Science*, pages 660–679. Springer-Verlag, 1994.

[Stå91] Gunnar Stålmarck. Normalization Theorems for Full First Order Classical Natural Deduction. *The Journal of Symbolic Logic*, 56(1):129–149, 1991.

[SU98] Morten H.B. Sørensen and Pawel Urzyczyn. The Curry-Howard Isomorphism. Lecture notes, Dept. of Computer Science, University of Copenhagen, 1998.

[Tar41] Alfred Tarski. On the Calculus of Relations. *The Journal of Symbolic Logic*, 6(3):73–89, 1941.

[TS96] A.S. Troelstra and H. Schwichtenberg. *Basic Proof Theory*. Cambridge Tracts in Theoretical Computer Science 43. Cambridge University Press, 1996.

[vB84] Johan van Benthem. Correspondence Theory. In Gabbay and Guenthner [GG84], pages 167–247.

[Ven90] Yde Venema. Expressiveness and Completeness of an Interval Tense Logic. *Notre Dame Journal of Formal Logic*, 31(4):529–547, 1990.

[Ven91] Yde Venema. A Modal Logic for Chopping Intervals. *Journal of Logic and Computation*, 1(4):453–476, 1991.

[Vig00] Luca Viganò. *Labelled Non-Classical Logics*. Kluwer Academic Publishers, 2000.

[Wal90] Lincoln A. Wallen. *Automated Proof Search in Non-Classical Logics* . MIT Press, 1990.

[Wan94] Heinrich Wansing. Sequent Calculi for Normal Modal Propositional Logics. *Journal of Logic and Computation*, 4(2):125–142, 1994.

[Yov97] Sergio Yovine. Kronos: A Verification Tool for Real-Time Systems. *International Journal of Software Tools for Technology Transfer* , 1(1/2):123–133, 1997.

[ZH98] Zhou Chaochen and Michael R. Hansen. An Adequate First Order Interval Logic. In *Compositionality, COMPOS'97*, volume 1536 of *Lecture Notes in Computer Science*, pages 584–608. Springer-Verlag, 1998.

[ZH01] Zhou Chaochen and Michael R. Hansen. Duration Calculus: A Formal Approach to Real-Time Systems. Unpublished draft, December 2001.

[ZHL95] Zhou Chaochen, Dang Van Hung, and Li Xiaoshan. A Duration Calculus with Infinite Intervals. In *Computation Theory, FCT'95*, volume 965 of *Lecture Notes in Computer Science*, pages 16–41. Springer-Verlag, 1995.

[ZHR91] Zhou Chaochen, C.A.R. Hoare, and Anders P. Ravn. A Calculus of Durations. *Information Processing Letters*, 40(5):269–276, 1991.

[ZHS93] Zhou Chaochen, Michael R. Hansen, and Peter Sestoft. Decidability and Undecidability Results for Duration Calculus. In *STACS'93*, volume 665 of *Lecture Notes in Computer Science*, pages 58–68. Springer-Verlag, 1993.

[ZL94] Zhou Chaochen and Li Xiaoshan. A Mean Value Calculus of Durations. In *A Classical Mind: Essays in Honour of C.A.R. Hoare* , pages 431–451. Prentice-Hall International, 1994.

[ZRH93] Zhou Chaochen, Anders P. Ravn, and Michael R. Hansen. An Extended Duration Calculus for Hybrid Systems. In *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 36–59. Springer-Verlag, 1993.

[ZZ94] Zheng Yuhua and Zhou Chaochen. A Formal Proof of the Deadline Driven Scheduler. In *Formal Techniques in Real-Time and Fault-Tolerant Systems, FTRTFT'94*, volume 863 of *Lecture Notes in Computer Science*, pages 756–775. Springer-Verlag, 1994.

[ZZYL94] Zhou Chaochen, Zhang Jingzhong, Yang Lu, and Li Xiaoshan. Linear Duration Invariants. In *Formal Techniques in Real-Time and Fault-Tolerant Systems, FTRTFT'94*, volume 863 of *Lecture Notes in Computer Science*, pages 86–109. Springer-Verlag, 1994.

[ØH95] Peter Øhrstrøm and Per F.V. Hasle. *Temporal Logic. From Ancient Ideas to Artificial Intelligence*. Kluwer Academic Publishers, 1995.

# Index