Three License Languages

From Domains to Script Design

Dines Bjørner[1,2,*], Arimoto Yasuhito[2], Chen Xiaoyi[2] and Xiang Jianwen[2,3]

[1] DTU Informatics, DK-2800 Kgs.Lyngby, Denmark;
[2] Graduate School of Information Science, Japan Advanced Institute of Science and Technology, 1-1 Asahidai, Tatsunokuchi, Nomi, Ishikawa 923-1292, Japan;
[3] Ubiquitous Software Group, Information Technology Research Institute, National Institute of Advanced Industrial Science and Technology, Akihabara Dai Bldg, 1-18-3 Sotokanda, Chiyoda-ku, Tokyo 101-0021, Japan

Summary. Classical digital rights license languages [2,11–13,19,21,24–32,35] were (and are) applied to the electronic "downloading", payment and rendering (playing) of artistic works (for example music, literature readings and movies). In this document we generalise such applications languages and we extend the concept of licensing to also cover work authorisation (work commitment and promises) in health care and in public government. The digital works for these two new application domains are patient medical records and public government documents.

Digital rights licensing for artistic works seeks to safeguard against piracy and to ensure proper payments for the rights to render these works. Health care and public government license languages seek to ensure transparent and professional (accurate and timely) health care, respectively 'good governance'. Proper mathematical definition of licensing languages seeks to ensure smooth and correct computerised management of licenses.

We shall motivate and exemplify three license languages, their pragmatics, syntax and informal as well as formal semantics.

1 Introduction

License:

a right or permission granted in accordance with law by a competent authority
to engage in some business or occupation,
to do some act, or to engage in some transaction
which but for such license would be unlawful

Merriam Webster On-line [38]

The concepts of licenses and licensing express relations between *actors* (licensors (the authority) and licensees), *simple entities* (artistic works, hospital

* Corresponding author: bjorner@gmail.com, www.imm.dtu/~db

patients, public administration and citizen documents) and *operations* (on simple entities), and as performed by actors. By issuing a license to a licensee, a licensor wishes to express and enforce certain permissions and obligations: which operations on which entities the licensee is allowed (is licensed, is permitted) to perform. In this paper we shall consider three kinds of entities: (i) digital recordings of artistic and intellectual nature: music, movies, readings ("audio books"), and the like, (ii) patients in a hospital as represented also by their patient medical records, and (iii) documents related to public government.

The *permissions* and *obligations* issues are, (i) for the owner (agent) of some intellectual property to be paid (i.e., an *obligation*) by users when they perform *permitted* operations (rendering, copying, editing, sub-licensing) on their works; (ii) for the patient to be professionally treated — by medical staff who are basically *obliged* to try to cure the patient; and (iii) for public administrators and citizens to enjoy good governance: transparency in law making (national parliaments and local prefectures and city councils), in law enforcement (i.e., the daily administration of laws), and law interpretation (the judiciary) — by agents who are basically *obliged* to produce certain documents while being *permitted* to consult (i.e., read, perhaps copy) other documents.

1.1 What Kind of Science Is This?

It is experimental computing science: The study and knowledge of how to design and construct software that is right, i.e., correct, and the right software, i.e., what the user wants. To study methods for getting the right software is interesting. To study methods for getting the software right is interesting. Domain development helps us getting the right software. Deriving requirements from domain descriptions likewise. Designing software from such requirements helps us get the software right. Understanding a domain and then designing license languages from such an understanding is new. We claim that computer-supported management of properly designed license languages is a hallmark of the e-Society.

1.2 What Kind of Contributions?

The experimental nature of the project being reported on is as follows: Postulate three domains. Describe these informally and formally. Postulate the possibility of license languages (LLs) that somehow relate to activities of respective domains. Design these – experimentally. Try discover similarities and differences between the three LLs (LL_{DRM}, LL_{HHLL}, LL_{PALL}). Formalise the common aspects: C_{LL}. Formalise the three LLs — while trying to "parameterise" the C_{LL} to achieve $LL_{DRM}, LL_{HHLL}, LL_{PALL}$. This investigation is bound to tell us something, we hope.

2 Pragmatics of The Three License Languages

- *By* **pragmatics** *we understand the study and practice of the factors that govern our choice of language in social interaction and the effects of our choice on others.*

In this section we shall rough-sketch-describe pragmatic aspects of the three domains of (i) production, distribution and consumption of artistic works (Sect. 2.1), (ii) the hospitalisation of patient, i.e., hospital health care (Sect. 2.2), and (iii) the handling of law-based document in public government (Sect. 2.3). The emphasis is on the pragmatics of the terms, i.e., the language used in these three domains.

2.1 The Performing Arts: Producers and Consumers

The intrinsic simple entities of the performing arts are the artistic works: drama or opera performances, music performances, readings of poems, short stories, novels, or jokes, movies, documentaries, newsreels, etc. We shall limit our span to the scope of electronic renditions of these artistic works: videos, CDs or other. In this paper we shall not touch upon the technical issues of "downloading"(whether "streaming" or copying, or other). That and other issues will be analysed in [41].

Operations on Digital Works

For a consumer to be able to enjoy these works that consumer must (normally first) usually "buy a ticket" to their performances. The consumer, i.e., the theatre, opera, concert, etc., "goer" (usually) cannot copy the performance (e.g., "tape it"), let alone edit such copies of performances. In the context of electronic, i.e., digital renditions of these performances the above "cannots" take on a new meaning. The consumer may copy digital recordings, may edit these, and may further pass on such copies or editions to others. To do so, while protecting the rights of the producers (owners, performers), the consumer requests permission to have the digital works transferred ("downloaded") from the owner/producer to the consumer, so that the consumer can render ("play") these works on own rendering devices (CD, DVD, etc., players), possibly can copy all or parts of them, then possibly can edit all or parts of the copies, and, finally, possibly can further license these "edited" versions to other consumers subject to payments to "original" licensor.

Past versus Future

In the past all a consumer of digital works could do was to download and possibly copy. We would like, in this document, to investigate what it might entail, with respect to a digital rights license language, to license the consumer to also (copy,) edit and sub-license such digital works.

License Agreement and Obligation

To be able to obtain these permissions the user agrees with the wording of some license and pays for the rights to operate on the digital works.

The Artistic Electronic Works: Two Assumptions

Two, related assumptions underlie the pragmatics of the electronics of the artistic works. The first assumption is that the format, the electronic representation of the artistic works is proprietary, that is, that the producer still owns that format. Either the format is publicly known or it is not, that is, it is somehow "secret". In either case we "derive" the second assumption (from the fulfilment of the first). The second assumption is that the consumer is not allowed to, or cannot operate[2] on the works by own means (software, machines). The second assumption implies that acceptance of a license results in the consumer receiving software that supports the consumer in performing all operations on licensed works, their copies and edited versions: rendering, copying, editing and sub-licensing.

Protection of the Artistic Electronic Works

The issue now is: how to protect the intellectual property (i.e., artistic) and financial (exploitation) rights of the owners of the possibly rendered, copied and edited works, both when, and when not further distributed.

An Artistic Digital Rights License Language

In Sects. 4.1, 5.1, and 5.2 we shall design a suitably flexible digital artistic works license language and, through its precise informal and formal description provide one set answers to the above issue.

2.2 Hospital Health Care: Patients and Medical Staff

Citizens go to hospitals in order to be treated for some calamity (disease or other), and by doing so these citizens become patients. At hospitals patients, in a sense, issue a request to be medically treated with the aim of full or partial restitution. This request is directed at medical staff, that is, the patient authorises medical staff to perform a set of actions upon the patient. One could claim, as we shall, that the patient issues a license.

Hospital Health Care: Patients and Patient Medical Records

So patients and their attendant patient medical records (PMRs) are the main entities, the "works" of this domain. We shall treat them synonymously: PMRs as surrogates for patients. Typical actions on patients — and hence on PMRs — involve admitting patients, interviewing patients, analysing patients, diagnosing patients, planning treatment for patients, actually treating patients, and, under normal circumstance, to finally release patients.

[2] render, copy and edit

Hospital Health Care: Medical Staff

Medical staff may request ('refer' to) other medical staff to perform some of these actions. One can conceive of describing action sequences (and 'referrals') in the form of hospitalisation (not treatment) plans. We shall call such scripts for licenses.

Professional Health Care

The issue is now, given that we record these licenses, their being issued and being honoured, whether the handling of patients at hospitals follow, or does not follow properly issued licenses.

A Hospital Health Care License Language

In Sects. 4.2, 5.1, and 5.3 we shall design a suitably flexible hospital health care license language and, through its precise informal and formal description provide one set answers to the above issue.

2.3 Public Government and the Citizens

The Three Branches of Government

By public government we shall, following Charles de Secondat, baron de Montesquieu (1689–1755)[3], understand a composition of three powers: the law-making (legislative), the law-enforcing and the law-interpreting parts of public government. Typically national parliament and local (province and city) councils are part of law-making government; law-enforcing government is called the executive (the administration, including the police and state and district attorneys); and law-interpreting government is called the judiciary (that is, judiciary system, which includes juries and judges etc.).

Documents

A crucial means of expressing public administration is through *documents*.[4]

We shall therefore provide a brief domain analysis of a concept of documents. (This document domain description also applies to patient medical records and, by some "light" interpretation, also to artistic works — insofar as they also are documents.)

Documents are *created*, *edited* and *read*; and documents can be *copied*, *distributed*, the subject of *calculations* (interpretations) and be *shared* and *shredded*.

[3] *De l'esprit des lois* (*The Spirit of the Laws*), published 1748

[4] Documents are, for the case of public government to be the "equivalent" of artistic works.

Document Attributes

With documents one can associate, as attributes of documents, the *actors* who initiate the following operations on documents: create, edit, read, copy, distribute (and to whom distributed), share, perform calculations and shred.

With these operations on documents, and hence as attributes of documents one can, again conceptually, associate the *location* and *time* of these operations.

Finally we shall associate with documents the following attributes: `(i) operations:` the name of operations that may be performed on the document; `(ii) actors:` the name of actors and which operations they may perform on the document; `(iii) time-stamped transactions and locations:` a chronological list of operations actually performed on the document and the location at which the document was placed at that time; etcetera. Many other attributes may be associatable. Please recall that we are "in the domain". This means that we can indeed talk about the above attributes being observable from documents. The documents may just be "good, old-fashioned" paper documents. But someone, some persons, knows, or recalls, or believes in the validity of such attributes or have stamp-marked or "asctibed" the documents in such a way that these attributes can be deduced. Appendix **??** (Pages **??**–**??**, which reflects [3]) presents "the beginnings" of "a theory of documents" in which these attribute assignments and observations can be done and made.

Actor Attributes and Licenses

With actors (whether agents of public government or citizens) one can associate the *authority* (i.e., the *rights*) these actors have with respect to performing actions on documents. We now intend to express these *authorisations as licenses*.

Document Tracing

An issue of public government is whether citizens and agents of public government act in accordance with the laws — with actions and laws reflected in documents such that the action documents enables a trace from the actions to the laws "governing" these actions.

We shall therefore assume that every document can be traced back to its law-origin as well as to all the documents any one document-creation or -editing was based on.

A Public Administration Document License Language

Sects. 4.3, 5.1, and 5.4 we shall design a suitably flexible public government document license language and, through its precise informal and formal description provide one set answers to the above issue.

3 Semantic Intents of Licensor and Licensee Actions

In this section we shall briefly analyse some of the common semantics of the three kinds of license languages that we intend to design.

3.1 Overview

There are two parties to a license: the *licensor* and the *licensee*. And there is a common agreement concerning a shared "item" between them, namely: the *license* and the *work item:* the artistic work, the patient, the document.

The licensor gives the licensee permission, or mandates the licensee to be obligated to perform certain actions on designated "items".

The licensee performs, or does not perform permitted and/or obligated actions

And the licensee may perform actions not permitted or not obligated.

The license shall serve to ensure that only permitted actions are carried out, and to ensure that all obligated actions are carried out.

Breach of the above, that is, breach of the contracted license may or shall result in revocation of the license.

3.2 Licenses and Actions

Licenses

Conceptually a licensor o (for owner) may issue a license named ℓ to licensee u (for user) to perform some actions. The license may syntactically appear as follows:

ℓ : **licensor** o **licenses licensee** u
to perform actions {a1,a2,...,an} **on work item** w

Actions

And, conceptually, the licensee (u) may perform actions which can be expressed as follows:

ℓ :a(w); ℓ :a′(w); ...; ℓ :a″(w); ...; ℓ :a‴(w)

These actions $(a, a', ..., a'', ..., a''')$ may be in the set {a1,a2,...,an}, mentioned in the license, or they may not be in that set. In the latter case we have a breach of license ℓ.

Two Languages

Thus there is *the language of licenses* and *the language of actions*.

We advise the reader to take note of the distinction between the permitted or obligated actions enumerated in a license and the license-name-labelled actions actually requested by a licensee.

3.3 Sub-licensing, Scheme I

A licensee u may wish to delegate some of the work associated with performing some licensed actions to a colleague (or customer). If, for example the license originally stated:

ℓ : **licensor** o **licenses licensee** u
to perform actions {a1,a2,...,an} **on work item** w

the licensee (u) may wish a colleague u' to perform a subset of the actions, for example

{ai,aj,...,ak} $\subseteq$ {a1,a2,...,an}

Therefore u would like if the above license

ℓ : **licensor** o **licenses licensee** u
to perform actions {a1,a2,...,an} **on work item** w

instead was formulated as:

ℓ : **licensor** o **licenses licensee** u
to perform actions {a1,a2,...,an} **on work item** w
allowing sub-licensing of actions {ai,aj,...,ak}

where

{ai,aj,...,ak} $\subseteq$ {a1,a2,...,an}

Now licensee u can perform the action

ℓ : **license actions** {a′,a″,a‴} **to** u′

where {a′,a″,a‴}$\subseteq${ai,aj,...,ak}.

The above is an action designator. Its practical meaning is that a license is issued by u:

$\eta(\ell$,u,t): **licensor** u **licenses licensee** u′
to perform actions {a′,a″,a‴} **on work item** w

The above license can be easily "assembled" from the action including the action named license: the context determines who (namely u) is issuing the license, and who or which is the work item. η is a function which applies to license name, agent identifications and time and yields unique new license names.

3.4 Sub-licensing, Scheme II

The subset relation

{ai,aj,...,ak} ⊆ {a1,a2,...,an}

mentioned in the sub-licensing part of license

ℓ : **licensor** o **licenses licensee** u
to perform actions {a1,a2,...,an} **on work item** w
allowing sub-licensing of actions {ai,aj,...,ak}

may be omitted. In fact one could relax the relation completely and allow any actions {ai,aj,...,ak} whether in {a1,a2,...,an} or not ! That is, the original licensor may mandate that a licensee allow a sub-licensee to perform operations that the licensee is not allowed to perform. Examples are: a licensee may break the shrink-wrap around some licensed software package — an action which may not be performed by the licensor; a medical nurse (i.e., a licensee) may perform actions on patients not allowed performed by the licensor (say, a medical doctor); and a civil servant (say, an archivist) may copy, distribute or shred documents, actions that may not be allowed by the licensor (i.e., the manager of that civil servant), while that civil servant (the archivist) is not allowed to create or read documents.

3.5 Multiple Licenses

Consider the following scenario: A licensee L is performing actions a_p, a_q, ..., a_r, on work item ω, and has licensed L' to perform actions a_i, a_j, ..., a_k, also on work item ω. The action whereby L licenses L' occurs at some time. At that time L has performed none or some of the actions a_p, a_q, ..., a_r (on work item ω), but maybe not all. What is going to happen? Can L and L' go on, in parallel, performing actions on the same work item (ω) ? Our decision is yes, and they can do so in an interleaved manner, not concurrently but alternating, i.e., not accessing the same work item simultaneously.

4 Syntax and Informal Semantics

We distinguish between the pragmatics, the semantics and the syntax of languages. Leading textbooks on (formal) semantics of programming languages are [15, 20, 33, 36, 39, 40].

We have already covered the concept of pragmatics and we have, in covering some basic issues of semantics, illustrated their application to some issues of license language design.

We shall now illustrate the use of syntax and semantics in license language design.

- *By* **syntax** *we mean (i) the ways in which words are arranged to show meaning (cf. semantics) within and between sentences, and (ii) rules for forming syntactically correct sentences.*
- *By* **semantics** *we mean the study and knowledge, including specification, of meaning in language [14].*
- By **informal** *semantics we mean a semantics which is expressed in concise natural language, for example, as here, English.*

4.1 A General Artistic License Language

We refer to the abstract syntax formalised below (that is, formulas 0.–8. [Page 10] and 9.–14. [Page 12]). The work on the specific form of the syntax has been facilitated by the work reported by Xiang JenWen [41].[5]

The Syntax

The syntax has two parts. One for licenses being issued by licensors. And one for the actions that licensees may wish to perform.

Licenses

We first present an abstract syntax of the language of licenses, then we annotate this abstract syntax, and finally we present an informal semantics of that language of licenses.

type
0. Ln, Nm, W, S, V
1. L = Ln × Lic
2. Lic == mkLic(licensor:Nm,licensee:Nm,work:W,cmds:Cmd-**set**)
3. Cmd == Rndr | Copy | Edit | RdMe | SuLi
4. Rndr = mkRndr(vw:(V|″`work`″),sl:S*)
5. Copy = mkCopy(fvw:(V|″`work`″),sl:S*,tv:V)
6. Edit = mkEdit(fvw:(V|″`work`″),sl:S*,tv:V)
7. RdMe = ″`readme`″
8. SuLi = mkSuLi(cs:Cmd-**set**,work:V)

Syntax Annotations

0: Syntax Sorts: (0.) Licenses are given names, ln:Ln, so are actors (owners, licensors, and users, licensees), nn:Nm. By w:W we mean a (net) reference to (a name of) the downloaded possibly segmented artistic work being licensed, and where segments are named (s:S), that is, s:S is a selector to either a segment of a downloaded work or to a segment of a copied and or and edited work.

[5] As this work, [41], has yet to be completed the syntax and annotations given here may change.

License Name and License Body: (1.) Every license (lic:Lic) has a unique name (ln:Ln). (2.) A license (lic:Lic) contains four parts: the name of the licensor, the name of the licensee, a reference to (the name of) the work and a set of command designators (that may be permitted to be performed on the work).

Commands: (3.) A command is either a render, a copy or an edit or a readme, or a sub-licensing (sub-license) command.

Render, Copy and Edit: (4.–6.) The render, copy and edit commands are each "decorated" with an ordered list of selectors (i.e., selector names) and a (work) variable name. The license command

copy ⟨s1,s7,s2⟩ v

means that the licensed work, ω, may have its sections s_1, s_7 and s_2 copied, in that sequence, into a new variable named v, Other copy commands may specify other sequences. Similarly for render and edit commands.

Read Me: (7.) The "readme" license command, in a license, ln, referring, by means of w, to work ω, somehow displays a graphical/textual "image" of information about ω. We do not here bother to detail what kind of information may be so displayed. But you may think of the following display information names of artistic work, artists, authors, etc., names and details about licensed commands, a table of fees for performing respective licensed commands, etcetera.

Sub-Licensing: (8.) The license command

license cmd1,cmd2,...,cmdn **on work** v
mkSuLi({cmd1,cmd2,...,cmdn},v)

means that the licensee is allowed to act as a licensor, to name sub-licensees (that is, licensees) freely, to select only a (possibly full) subset of the sub-licensed commands (that are listed) for the sub-licensees to enjoy. The license need thus not mention the name(s) of the possible sub-licensees. But one could design a license language, i.e., modify the present one, to reflect such constraints. The license also does not mention the payment fee component. As we shall see under licensor actions such a function will eventually be inserted.

Informal Semantics

A license licenses the licensee to render, copy, edit and license (possibly the results of editing) any selection of downloaded works. In any order — but see below — and any number of times. For every time any of these operations take place payment takes place according to the payment function (which can be inspected by means of the ``readme'' command). The user can render the downloaded work and can render copies of the work as well as edited versions of these. Edited versions are given own names. Editing is always of copied

versions. Copying is either of downloaded or of copied or edited versions. This does not transpire from the license syntax but is expressed by the licensee, see below, and can be checked and duly paid for according to the payment function.

The payment function is considered a partial function of the selections of the work being licensed.

Please recall that licensed works are proprietary. Either the work format is known, or it is not supposed to be known, In any case, the rendering, editing, copying and the license-"assembling" functions (see next section) are part of the license and the licensed work and are also assumed to be proprietary. Thus the licensee is not allowed to and may not be able to use own software for rendering, editing, copying and license assemblage.

Licenses specify sets of permitted actions. Licenses do not normally mandate specific sequences of actions. Of course, the licensee, assumed to be an un-cloned human, can only perform one action at a time. So licensed actions are carried out sequentially. The order is not prescribed, but is decided upon by the licensee. Of course, some actions must precede other actions. Licensees must copy before they can edit, and they usually must edit some copied work before they can sub-license it. But the latter is strictly speaking not necessary.

Actions

type
9. V
10. Act = Ln × (Rndr|Copy|Edit|License)
11. Rndr == mkR(sel:S*,wrk:(W|V))
12. Copy == mkC(sel:S*,wrk:(W|V),into:V)
13. Edit == mkE(wrks:V*,into:V)
14. License == mkL(ln:Ln,licensee:Nm,wrk:V,cmds:Cmd-**set**,fees:PF)

Annotations and Informal Semantics:

Variables: (9.) By V we mean the name of a variable in the users own storage into which downloaded works can be copied (now becoming a local work). The variables are not declared. They become defined when the licensee names them in a copy command. They can be overwritten. No type system is suggested.

Actions: (10.) Every action of a licensee is tagged by the name of a relevant license; if the action is not authorised by the named license then it is rejected; render and copy actions mention a specific sequence of selectors; and if this sequence is not an allowed (a licensed) one, then the action is rejected. (Notice that the license may authorise a specific action, *a* with different sets of sequences of selectors — thus allowing for a variety of possibilities as well as constraints.)

Render: (11.) The licensee, having now received a license, can render selections of the licensed work, or of copied and/or edited versions of the licensed work. No reference is made to the payment function. When rendering, the semantics is that this function is invoked and duly applied. That is, render payments are automatically made: subtracted from the licensees account and forwarded to the licensor.

Copy: (12.) The licensee can copy selections of the licensed work, or of previously copied and/or edited versions of the licensed work. The licensee identifies a name for the local storage file where the copy will be kept. No reference is made to the payment function. When copying, the semantics is that this function is invoked and duly applied. That is, copy payments are automatically made: subtracted from the licensees account and forwarded to the licensor.

Edit: (13.) The licensee can edit selections of the licensed work, or of copied and/or previously edited versions of the licensed work. The licensee identifies a name for the local storage file where the new edited version will be kept. The result of editing is a new work. No reference is made to the payment function. When editing, the semantics is that this function is invoked and duly applied. That is, edit payments are automatically made: subtracted from the licensees account and forwarded to the licensor.

(a) Although no reference is made to any edit functions these are made available to the licensee when invoking the edit command. (b) You may think of these edit functions being downloaded at the time of downloading the license. (c) Other than this we need not further specify the editing functions.

Remarks (a,b,c) apply also to the above copying function.

Sub-Licensing: (14.) The licensee can further sub-license copied and/or edited work. The licensee must give the license being assembled a unique name. And the licensee must choose to whom to license this work. A sub-license, like does a license, authorises which actions can be performed, and then with which one of a set of alternative selection sequences. No payment function is explicitly mentioned. It is to be semi-automatically derived (from the originally licensed payment fee function and the licensee's payment demands) by means of functionalities provided as part of the licensed payment fee function.

The sub-license command information is thus compiled (assembled) into a license of the form given in (1.–3.) and schematised in the "$\eta(\ell, u, t)$:" labelled command designator on Page 8. The licensee becomes the licensor and the recipient of the new, the sub-license, become the new licensee. The assemblage refers to the context of the action. That context knows who, the licensor, is issuing the sub-license. From the license label of the command it is known whether the sub-license actions are a subset of those for which sub-licensing has been permitted.

4.2 A Hospital Health Care License Language

A reading of this section assumes that of having read the previous section.

We refer to the abstract syntax formalised below (that is, formulas 0.–7. Page 15). The work on the specific form of the syntax has been facilitated by the work reported in [1].[6]

A Notion of License Execution States

In the context of the Artistic License Language licensees could basically perform licensed actions in any sequence and as often as they so desired. There were, of course, some obvious constraints. Operations (not to be confused with hospital surgery on patients) on local works could not be done before these had been created — say by copying. Editing could only be done on local works and hence required a prior action of, for example, copying a licensed work. In the context of hospital health care most of the actions can only be performed if the patient has reached a suitable state in the hospitalisation. We refer to Fig. 1 for an idealised hospitalisation plan.

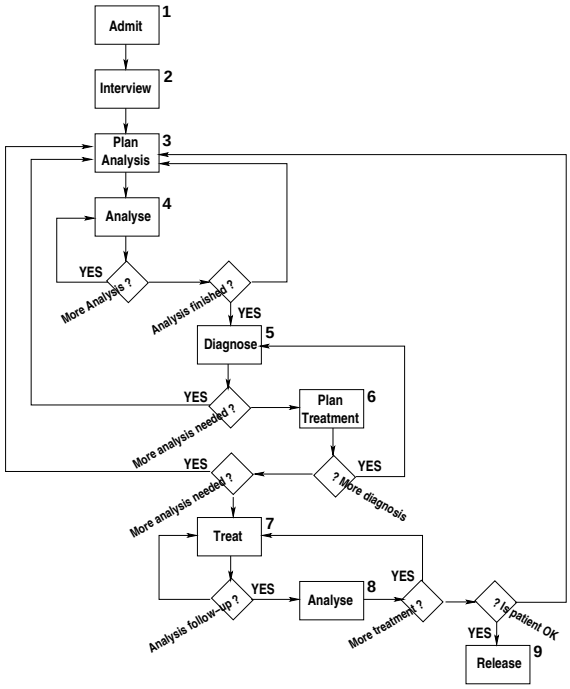

Fig. 1. An example hospitalisation plan. States: {1,2,3,4,5,6,7,8,9}

[6] As this work, [1], has yet to be completed the syntax and annotations given here may change.

We therefore suggest to join to the licensed commands an indicator which prescribe the (set of) state(s) of the hospitalisation plan in which the command action may be performed.

Two or more medical staff may now be licensed to perform different (or even same!) actions in same or different states. If licensed to perform same action(s) in same state(s) — well that may be "bad license programming" if and only if it is bad medical practice! One cannot design a language and prevent it being misused!

The Syntax

The syntax has two parts. One for licenses being issued by licensors. And one for the actions that licensees may wish to perform.

Licenses

type
0. Ln, Mn, Pn
1. License = Ln × Lic
2. Lic == mkLic(s_staff1:Mn,s_mandate:ML,s_pat:Pn)
3. ML == mkML(s_staff2:Mn,s_to_perform_acts:CoL**-set**)
4 CoL = Cmd | ML | Alt
5. Cmd == mkCmd(s_σs:Σ**-set**,s_stmt:Stmt)
6 Alt == mkAlt(cmds:Cmd**-set**)
7. Stmt = **admit** | **interview** | **plan-analysis** | **do-analysis**
| **diagnose** | **plan-treatment** | **treat** | **transfer** | **release**

The above syntax is correct RSL [4–7, 16–18]. But it is decorated! The sub-types {|**boldface keyword**|} are inserted for readability.

Syntax: Annotations: (0.) Licenses, medical staff and patients have names.

(1.) Licenses further consist of license bodies (Lic).

(2.) A license body names the licensee (Mn), the patient (Pn), and,

(3.) through the "mandated" licence part (ML), it names the licensor (Mn) and which set of commands (C) or (o) implicit licenses (L, for a "total" mnemonic identifier: CoL) the licensor is mandated to issue.

(4.) An explicit command or licensing (CoL) is either a command (Cmd), or a sub-license (ML) or an alternative (Alt).

(5.) A command (Cmd) is a set-of-states-labelled statement.

(3.) A sub-license (ML) just states the command set that the sub-license licenses. As for the Artistic License Language the licensee chooses an appropriate subset of commands. The context "inherits" the name of the patient. But the sub-licensee is explicitly mandated in the license!

(6.) An alternative (Alt) is also just a set of commands. The meaning is that either the licensee choose to perform the designated actions or, as for ML, but now freely choosing the sub-licensee, the licensee (now new licensor) chooses to confer actions to other staff.

(7.) A statement (Stmt) is either an admit, an interview, a plan analysis, an analysis, a diagnose, a plan treatment, a treatment, a transfer, or a release directive. Information given in the patient medical report, for the designated state, inform medical staff as to the details of analysis, what to base a diagnosis on, of treatment, etc.

Actions

8. Action = Ln × Act
9. Act = Stmt | SubLic
10. SubLic = mkSubLic(sublicensee:Ln,license:ML)

Syntax Annotations: (8.) Each action actually attempted by a medical staff refers to the license, and hence the patient name.

(9.) Actions are either of an admit, an interview, a plan analysis, an analysis, a diagnose, a plan treatment, a treatment, a transfer, or a release actions. Each individual action is only allowed in a state σ if the action directive appears in the named license and the patient (medical record) designated state set σs.

(10.) Or an action can be a sub-licensing action. Either the sub-licensing action that the licensee is attempting is explicitly mandated by the license (4. ML), or is an alternative one thus implicitly mandated (6.). The full sub-license, as defined in (1.–3. on Page 15) is compiled from contextual information.

Informal Semantics

An informal, rough-sketch semantics (here abbreviated) would state that a prescribed action is only performed if the patient, cum patient medical record is in an appropriate state; that the patient is being treated according to the action performed; and that records of this treatment and its (partially) analysed outcome is introduced into the patient medical record. The next state of the patient, cum patient medical record, depends on the outcome of the treatment[7]; and hence the patient medical record carries with it, i.e., embodies a, or the, hospitalisation plan in effect for the patient, and a reference to the current state of the patient.

4.3 A Public Administration License Language

In this appendix we shall assume that the reader has studied, or at least can refer to Appendix **??** (Pages **??**–**??**, [3]).

We refer to the abstract syntax formalised below (that is, formulas (0.–19.), Page 17 and formulas (20.–31.), starting Page 19). The work on the specific form of the syntax has been facilitated by the work reported in [3, 8, 10].[8]

[7] Cf. the diamond-shaped decision boxes in Fig. 1 on page 14.

[8] As part this work, [10], has yet to be completed the syntax and annotations given here may change.

The Syntax: Licenses

The syntax has two parts. One for licenses being issued by licensors. And one for the actions that licensees may wish to perform.

The Form of Licenses

type
0. Ln, An, Dn, DCn, Cfn
1. L == Grant | Extend | Restrict | Withdraw
2. Grant == mkG(s_license:Ln,s_licensor:An,s_ops:OpDocs,s_licensee:An)
3. Extend == mkE(s_licensor:An,s_licensee:An,s_license:Ln,s_with_ops:OpDocs)
4. Restrict == mkR(s_licensor:An,s_licensee:An,s_license:Ln,s_to_ops:OpDocs)
5. Withdraw == mkW(s_licensor:An,s_licensee:An,s_license:Ln)
6a. OpDocs = Op $\overrightarrow{m}$ (Dn $\overrightarrow{m}$ DCn)
6b. Op == Crea|Edit|Read|Copy|Licn|Shar|Rvok|Rlea|Rtur|Calc|Shrd

Licensed Operations

type
7. UDI
8. Crea == mkCr(s_dn:Dn,s_doc_class:DCn,s_based_on:UDI-**set**)
9. Edit == mkEd(s_doc:UDI,s_based_on:UDI-**set**)
10. Read == mkRd(s_doc:UDI)
11. Copy == mkCp(s_doc:UDI)
12. Licn == mkLi(s_kind:LiTy)
13. LiTy == grant | extend | restrict | withdraw
14. Shar == mkSh(s_doc:UDI,s_with:An-**set**)
15. Rvok == mkRv(s_doc:UDI,s_from:An-**set**)
16. Rlea == mkRl(s_dn:Dn)
17. Rtur == mkRt(s_dn:Dn)
18. Calc == mkCa(s_fcts:CFn-**set**,s_docs:UDI-**set**)
19. Shrd == mkSh(s_doc:UDI)

Syntax & Informal Semantics Annotations: Licenses

(0.) The are names of licenses (Ln), actors (An), documents (Dn), document classes (DCn), calculation functions (Cfn).

(1.) There are four kinds of licenses: granting, extending, restricting and withdrawing.

(2.) Actors (licensors) grant licenses to other actors (licensees). An actor is constrained to always grant distinctly named licenses. No two actors grant identically named licenses.[9] A set of operations on named documents (OpDocs) are granted.

[9] This constraint can be enforced by letting the unique actor name be part of the license name.

(3.–5.) Actors who have issued named licenses may extend, restrict or withdraw the license rights (wrt. operations).

(6a.) To each granted operation there is associated a set of document names (each of which is associated with a document class name, i.e., a type).

(6b.) There are nine kinds of operation (Op) authorisations.

Some of the next explications also explain parts of some of the corresponding actions (see (20.–32.) Page 19).

(7.) There are unique document identifiers. Several documents may be "of the same" name, but each created document "of that name" is ascribed a unique document identifier.

(8.) Creation results in an initially void document which is not necessarily uniquely named (dn:Dn) (but that name is associated with the unique document identifier created when the document is created):

value
obs_Dn: UID → Dn, obs_DCn: UID → DCn, obs_An: UID → An

The created document is typed by a document class name (dcn:DCn) which, like the name of the licensee, can also be observed from the unique document identifier). The created document is possibly based on[10] one or more identified documents (over which the licensee (at least) has reading rights). We can presently omit consideration of the document class concept. The "based on" documents are moved[11] from licensor to licensee.

(9.) Editing a document may be based on "inspiration" from, that is, with reference to a number of other documents (over which the licensee (at least) has reading rights). What this "be based on" means is simply that the edited document contains those references. (They can therefore be traced.) The "based on" documents are moved[12] from licensor to licensee — if not already so moved as the result of the specification of other authorised actions.

(10.) Reading a document only changes its "having been read" status (etc.) — as per Appendix **??** [3]. The read document, if not the result of a copy, is moved from licensor to licensee — if not already so moved as the result of the specification of other authorised actions.

(11.) Copying a document increases the document population by exactly one document. All previously existing documents remain unchanged except that the document which served as a master for the copy has been so marked. The copied document is like the master document except that the copied document is marked to be a copy (etc.) — as per Appendix **??** [3]. The master document, if not the result of a create or copy, is moved from licensor to licensee — if not already so moved as the result of the specification of other authorised actions.

[10] "Based on" means that the initially void document contains references to those (zero, one or more) documents.They can therefore be traced (etc.) — as per [3].

[11] A discussion on this choice, of "move", should weigh this against licensee be able to remotely access the "based on" document, etc., etc.

[12] See Footnote 10.

(12.) A licensee can sub-license certain operations to be performed by other actors.

(13.) The granting, extending, restricting or withdrawing permissions (a) cannot name a license (the user has to do that); (b) do not need to refer to the licensor (the licensee issuing the sub-license); and (c) leaves it open to the licensor to freely choose a licensee. One could, instead, for example, constrain the licensor to choose from a certain class of actors. The licensor (the licensee issuing the sub-license) must choose a unique license name.

(14.) A document can be shared between two or more actors. One of these is the licensee, the others are implicitly given read authorisations. (One could think of extending, instead, the licensing actions with a **shared** attribute.) The shared document, if not the result of a create and edit or copy, is moved from licensor to licensee — if not already so moved as the result of the specification of other authorised actions. Sharing a document does not move nor copy it.

(15.) Sharing documents can be revoked. That is, the reading rights are removed.

(16.) The release operation: if a licensor has authorised a licensee to create a document (and that document, when created got the unique document identifier udi:UDI) then that licensee can release the created, and possibly edited document (by that identification) to the licensor, say, for comments. The licensor thus obtains the master copy.

(17.) The return operation: if a licensor has authorised a licensee to create a document (and that document, when created got the unique document identifier udi:UDI) then that licensee can return the created, and possibly edited document (by that identification) to the licensor — "for good"! The licensee relinquishes all control over that document.

(18.) One or more documents can be subjected to any one of a set of permitted calculation functions. These documents, if not the result of creates and edits or copies, are moved from licensor to licensee — if not already so moved as the result of the specification of other authorised actions. Observe that there can be many calculation permissions, over overlapping documents and functions.

(19.) A document can be shredded.

The Syntax: Actions

type
20. Action = Ln × Clause
21. Clause = Cre | Edt | Rea | Cop | Lic | Sha | Rvk | Rel | Ret | Cal | Shr
22. Cre == mkCre(dcn:DCn,based_on_docs:UID-**set**)
23. Edt == mkEdt(uid:UID,based_on_docs:UID-**set**)
24. Rea == mkRea(uid:UID)
25. Cop == mkCop(uid:UID)
26. Lic == mkLic(license:L)

27. Sha == mkSha(uid:UID,with:An-**set**)
28. Rvk == mkRvk(uid:UID,from:An-**set**)
29. Rel == mkRel(dn:Dn,uid:UID)
30. Ret == mkRet(dn:Dn,uid:UID)
31. Cal == mkCal(fct:Cfn,over_docs:UID-**set**)
32. Shr == mkShr(uid:UID)

Preliminary Remarks

A clause elaborates to a state change and usually some value. The value yielded by elaboration of the above create, copy, and calculation clauses are unique document identifiers. These are chosen by the "system".

Syntax & Informal Semantics Annotations: Actions

(20.) Actions are tagged by the name of the license with respect to which their authorisation and document names has to be checked. No action can be performed by a licensee unless it is so authorised by the named license, both as concerns the operation (create, edit, read, copy, license, share, revoke, calculate and shred) and the documents actually named in the action. They must have been mentioned in the license, or, created or copies of downloaded (and possibly edited) documents or copies of these — in which cases operations are inherited.

(22.) A licensee may create documents if so licensed — and obtains all operation authorisations to this document.

(23.) A licensee may edit "downloaded" (edited and/or copied) or created documents.

(24.) A licensee may read "downloaded" (edited and/or copied) or created and edited documents.

(25.) A licensee may (conditionally) copy "downloaded" (edited and/or copied) or created and edited documents. The licensee decides which name to give the new document, i.e., the copy. All rights of the master are inherited to the copy.

(26.) A licensee may issue licenses of the kind permitted. The licensee decides whether to do so or not. The licensee decides to whom, over which, if any, documents, and for which operations. The licensee looks after a proper ordering of licensing commands: first grant, then sequences of zero, one or more either extensions or restrictions, and finally, perhaps, a withdrawal.

(27.) A "downloaded" (possibly edited or copied) document may (conditionally) be shared with one or more other actors. Sharing, in a digital world, for example, means that any edits done after the opening of the sharing session, can be read by all so-granted other actors.

(28.) Sharing may (conditionally) be revoked, partially or fully, that is, wrt. original "sharers".

(29.) A document may be released. It means that the licensor who originally requested a document (named dn:Dn) to be created is now being able to see the results — and is expected to comment on this document and eventually to re-license the licensee to further work.

(30.) A document may be returned. It means that the licensor who originally requested a document (named dn:Dn) to be created is now given back the full control over this document. The licensee will no longer operate on it.

(31.) A license may (conditionally) apply any of a licensed set of calculation functions to "downloaded" (edited, copied, etc.) documents, or can (unconditionally) apply any of a licensed set of calculation functions to created (etc.) documents. The result of a calculation is a document. The licensee obtains all operation authorisations to this document (— as for created documents).

(32.) A license may (conditionally) shred a "downloaded" (etc.) document.

4.4 Discussion

Comparisons

We have "designed", or rather proposed three different kinds of license languages. In which ways are they "similar", and in which ways are they "different"? Proper answers to these questions can only be given after we have formalised these languages. The comparisons can be properly founded on comparing the semantic values of these languages.

But before we embark on such formalisations we need some informal comparisons so as to guide our formalisations. So we shall attempt such analysis now with the understanding that it is only a temporary one.

Differences

Work Items: The work items of the artistic license language(s) are essentially "kept" by the licensor. The work items of the hospital health care license language(s) are fixed and, for a large set of licenses there is one work item, the patient which is shared between many licensors and licenses. The work items of the public administration license language(s) — namely document — are distributed to or created and copied by licenses and may possibly be shared.

Operations: The operations of the artistic license language(s) are are essentially "kept" by the licensor. The operations of the hospital health care license language(s) are are essentially "kept" by the licensees (as reflected in their professional training and conduct). The operations of the public administration license language(s) are essentially "kept" by the licensees (as reflected in their professional training and conduct).

Permissions and Obligations: Generally we can say that the modalities of the artistic license language(s) are essentially permissions with **payment** (as well as use of licensor functions) being an obligation; that the modalities of the hospital health care license language(s) are are essentially obligations to maintain professional standards and the Hippocractic oath; and, as well, that the modalities of the public administration license language(s) are essentially obligations to maintain professional and legal standards. We may have more to say about permissions and obligations later.

5 Formal Semantics

By formal semantics we understand a definition expressed in a formal language, that is, a language with a mathematical syntax, a mathematical semantics, and a consistent and relative complete proof system. We shall initially deploy the CSP [22, 23, 34, 37] Specification Language embedded in a Landin–like notation of **let** clauses[13]. This embedding is expressed, as were the syntaxes, in the RAISE Specification Language, RSL [4–7, 16–18].

5.1 A Model of Common Aspects

Actors: Behaviours and Processes

We see the system as a set of actors. An actor is either a licensor, or a licensee, or, usually, such as we have envisaged our license languages, both. To each actor we associate a behaviour — and we model actor behaviours as CSP processes. So the system is then modelled as a set of concurrent behaviours, that is, parallel (‖) CSP processes. Actors are uniquely identified (Aid).

System States

With each actor behaviour we associate a state (ω:Ω). "Globally" initial such state components are modelled as maps from actor identifiers to states. We shall later analyse these states into major components.

type
  Aid, Ω
  Ωs = Aid $\xrightarrow[m]{}$ Ω

[13] — known since the very early 1960's

System Processes

Actor processes communicate with one another over channels. There is a set of actor identifier indexed channels. The channels carry actor name decorated messages (Aid×M). Potential licensees request licenses. Licensors issue licenses in response to requests. Work items are communicated over these channels. As are payments and reports on use of licensed operations on licensed work items. So there is a large variery of messages. An actor is either pro-active, requesting licenses, sending payment or reports, or re-active: responding to license requests, sending work items. An actor non-deterministically (⌈⌉) alternates between these activities.

```
type
  M = Lic | Pay | Rpt | ...
channel
  {a[ i ]|i:Aid} (Aid×M)
value
  actor: j:Aid × Ω → in,out {a[ i ]|i:Aid•i≠j} Unit
  actor(j)(ω) ≡ let ω′ = pro_act(j)(ω)⌈⌉re_act(j)(ω) in actor(j)(ω′) end

  system: Ωs → Unit
  system(ωs) ≡ ‖ {actor(i)(ωs(i))|i:Aid}
```

Actor Processes

We have identified two kinds of actor processes: (a) pro-active, during which the actor, by own initiative, (as a prospective licensee) may request a license from a prospective licensor, or, as an actual licensee, and as the result of performing licensed actions, sends payments or reports (or other) to the licensor; and (b) re-active, during which the actor, in response to requests (as a licensor) sends a requesting actor a license (whereby the requester becomes a licensee), or "downloads" (access to) requested works or functions. functions.

The Pro-active Actor Behaviour: In the pro-active behaviour an actor, (1.), at will, i.e., (2.) non-deterministically internal choice (⌈⌉), decides to either request a license (rl) or to perform some action (op). In the former case the actor inquires (4., l_iq) an own state to find out from which licensor (k), and which kind of license requirements (l_rq) is to be requested. This licensor and these requirements are duly noted in the state (ω'). After (5.) sending the request the actor continues being an actor in the duly noted state (ω'). In the latter case (6., op) there may be many "next" actions to do. The actor inquires (a_iq) an own state (ω) to find out which action (op_i) is "next". The actor then (7.) performs (act) the designated operation. It is here, for simplification assumed that all operation completions imply a "completion" message (m: a payment, a report, or other) to the operation licensing actor (k). So such a message is sent (8.) and the operation-updated local state (ω') is yielded — whereby the pro-active actor "resumes" being an actor as from that state.

```
type
   M = Lic | Pay | Rpt | ...
channel
   {a[ i ]|i:Aid} (Aid×M)
value
    pro_act: j:Aid → Ω → in,out {a[ i ]|i:Aid•j≠i}  Ω
1.  pro_act(j)(ω) ≡
2.    let what_to_do = rl ⌈⌉ op in
3.    case what_to_do of
4.      rl → let (k,l_rq,ω′)=iq_l_Ω(ω) in
5.           a(k)!(j,l_rq);ω′ end
6.      op → let op_i=iq_a_Ω(ω) in
7.           let (k,m,ω′)=act(op_i)(ω) in
8.           a(k)!(j,m) ; ω′ end end
      end end
```

The Re-active Actor Behaviour: In the re-active behaviour an actor (9., j), is willing to engage in communication with other actors. This is formalised by a non-deterministic external choice (10., ⌈⌉) between either of a set ({...}) of (zero, or more) other actors (k:Aid\{j}) who are trying to contact the re-active actor. The communicated message (k,m) reveals the identity (k) of the requesting, i.e., the pro-active actor,[14] The message, m, reveals what action (act(m)(ω)) the re-active actor is requested to perform. The actor does so (11.). This results in a reply message m′ and a state change (ω′). The reply message (a(k)!(j,m′)) is sent (12.) to the requesting actor (k); and the re-active actor (j) yields the requested action-updated state (ω′) — whereby the re-active actor "resumes" being an actor as from that state.

```
type
   M = Lic | Pay | Rpt | ...
channel
   {a[ i ]|i:Aid} (Aid×M)
value
   re_act: j:Aid → Ω → in,out {a[ i ]|i:Aid•j≠i}  Ω
9.  re_act(j)(ω)≡
10.   let (k,m)=⌈⌉{a(k)?|k:Aid} in
11.   let (m′,ω′)=act(m)(ω) in
12.   a(k)!(j,m′);ω′ end end
```

[14] Do not get confused by the two k's on either side of the let clause. The left k is yielded by the (input) communication a(k)?. The defining scope of the right side k, as in a(k), is just the right-hand side of the left clause.

Functions

We first list (and "read") the signatures of the two auxiliary (iq_l_Ω, iq_a_Ω) and one elaboration (act) function assumed in the definition of the pro- and reactive actor processes. After that we discuss the former and suggest definitions of the latter.

Auxiliary Function Signatures

The inquire license function (iq_l_Ω) inspects the actor's state to ("eureka") determine which most desirable licensor (k:Aid) offers which one kind of desired license requirements (License_Requirements). The inquire action function (iq_a_Ω) inspects the actor's state to (somewhat "eureka") determine which action is "next" to be performed. That action is being designated (Action_Designator).

type
  License_Requirements,Action_Designation
value
  iq_l_Ω: $\Omega \to$ Aid $\times$ License_Requirements $\times$ Ω
  iq_a_Ω: $\Omega \to$ Action_Designator

By 'eureka'[15] is meant that the inquiry is internal non-deterministic, that is, is not influenced by an outside, could have any one of very many outcomes, and can thus only be rather loosely defined.

Elaboration Function Signature

The action performing function (act) "finds" the designated operation in the current state, applies it in the current state, and yields ("read" backwards) a possibly new state ($\omega : \Omega$), a message (m:M) to be sent to the licensor (k:Aid) who authorised the operation and may need or which to have a payment, a report, or some such thing "back"!

type
  Action_Designation
value
  act: Action_Designation $\to \Omega \to$ Aid $\times$ M $\times$ Ω

Discussion of Auxiliary Functions

The auxiliary functions are usually not computable functions. The actors are not robots. And it is not necessary to further define these functions beyond stating their signatures as they are usually performed by human actors. The signature of the inquire license function expresses a possible change to the

[15] "Eureka" used to express triumph on a discovery, heuristics

inquired state. One would think of the inquiring actor somehow noting down, remembering, as it were, which inquiries were attempted or had been made. The signature of the inquire actions function does not express such a state change. But it could be expressed as well.

Schema Definitions of Elaboration Functions

5.2 A Model of The General Artistic License Language

A Licensor/Licensee State

Auxiliary Functions

Elaboration Functions

5.3 A Model of The Hospital Health Care License Language

A Patient [Medical Record] State

A Medical Staff State

Auxiliary Functions

Elaboration Functions

5.4 A Model of The Public Administration License Language

A Public Administrator State

Auxiliary Functions

Elaboration Functions

5.5 Discussion

6 Conclusion

It is too early — in the development of this report — to conclude!

6.1 Summary: What Have We Achieved?

Or rather, at this early, incomplete stage, what do we wish to achieve? In a first round we wish to achieve the following: an understanding of different kinds of license languages; an understanding of obligations and permissions (yet to be "designed" more explicitly into the three languages; a formalisation of both common aspects of the license systems (as a "vastly" distributed set of very many actors acting on even more licenses "competing" for resources, etc.), as well as of each individual language.

6.2 Open Issues

An Open Issue: Rôle of Modal Logics

Temporal logic Deontic logic (permission and obligation) Logic of knowledge and belief etc.

Another Open Issue: Fair Use

Fair use: ... etc.

6.3 Acknowledgements

The second author wishes to express that it has been an interesting experience to work with his three co-author JAIST "students": Mr. Yasuhito Arimoto, MSc student, Miss Chen Xiaoyi, PhD student, and Dr. Xiang Jianwen, Post-doc.

7 Bibliographical Notes

References

1. Yasuhito Arimoto and Dines Bjørner. Hospital Healthcare: A Domain Analysis and a License Language. Technical note, JAIST, School of Information Science, 1-1, Asahidai, Tatsunokuchi, Nomi, Ishikawa, Japan 923-1292, Summer 2006.
2. Alapan Arnab and Andrew Hutchison. Fairer Usage Contracts for DRM. In *Proceedings of the Fifth ACM Workshop on Digital Rights Management (DRM'05)*, pages 65–74, Alexandria, Virginia, USA, Nov 2005.
3. Dines Bjørner. Documents: A Domain Analysis. Technical note, JAIST, School of Information Science, 1-1, Asahidai, Tatsunokuchi, Nomi, Ishikawa, Japan 923-1292, Summer 2006.
4. Dines Bjørner. *Software Engineering, Vol. 1: Abstraction and Modelling*. Texts in Theoretical Computer Science, the EATCS Series. Springer, 2006.
5. Dines Bjørner. *Software Engineering, Vol. 2: Specification of Systems and Languages*. Texts in Theoretical Computer Science, the EATCS Series. Springer, 2006. Chapters 12–14 are primarily authored by Christian Krog Madsen.
6. Dines Bjørner. *Software Engineering, Vol. 3: Domains, Requirements and Software Design*. Texts in Theoretical Computer Science, the EATCS Series. Springer, 2006.
7. Dines Bjørner. *Software Engineering, Vol. I: The Triptych Approach, Vol. II: A Model Development*. To be submitted to Springer for evaluation, expected published 2009. This book is the basis for guest lectures at Techn. Univ. of Graz, Politecnico di Milano, University of the Saarland (Germany), etc., 2008–2009.
8. Dines Bjørner and XiaoYi Chen. Public Government: A Domain Analysis. Technical note, JAIST, School of Information Science, 1-1, Asahidai, Tatsunokuchi, Nomi, Ishikawa, Japan 923-1292, Summer 2006.

9. Dines Bjørner and Martin C. Henson, editors. *Logics of Specification Languages — see [?, ?, ?, ?, ?, ?, ?, ?, ?, 16].* EATCS Monograph in Theoretical Computer Science. Springer, Heidelberg, Germany, 2008.
10. XiaoYi Chen and Dines Bjørner. Public Government: A Domain Analysis and a License Language. Technical note, JAIST, School of Information Science, 1-1, Asahidai, Tatsunokuchi, Nomi, Ishikawa, Japan 923-1292, Summer 2006.
11. C. N. Chong, R. J. Corin, J. M. Doumen, S. Etalle, P. H. Hartel, Y. W. Law, and A. Tokmakoff. LicenseScript: a logical language for digital rights management. *Annals of telecommunications special issue on Information systems security*, 2006.
12. C. N. Chong, S. Etalle, and P. H. Hartel. Comparing Logic-based and XML-based Rights Expression Languages. In *Confederated Int. Workshops: On The Move to Meaningful Internet Systems (OTM)*, number 2889 in LNCS, pages 779–792, Catania, Sicily, Italy, 2003. Springer.
13. Cheun Ngen Chong, Ricardo Corin, and Sandro Etalle. LicenseScript: A novel digital rights languages and its semantics. In *Proc. of the Third International Conference WEB Delivering of Music (WEDELMUSIC'03)*, pages 122–129. IEEE Computer Society Press, 2003.
14. David Crystal. *The Cambridge Encyclopedia of Language.* Cambridge University Press, 1987, 1988.
15. J.W. de Bakker. *Control Flow Semantics.* The MIT Press, Cambridge, Mass., USA, 1995.
16. Chris George and Anne E. Haxthausen. *Logics of Specification Languages*, chapter The Logic of the RAISE Specification Language, pages 349–399 in [9]. Springer, 2008.
17. Chris W. George, Peter Haff, Klaus Havelund, Anne Elisabeth Haxthausen, Robert Milne, Claus Bendix Nielsen, Søren Prehn, and Kim Ritter Wagner. *The RAISE Specification Language.* The BCS Practitioner Series. Prentice-Hall, Hemel Hampstead, England, 1992.
18. Chris W. George, Anne Elisabeth Haxthausen, Steven Hughes, Robert Milne, Søren Prehn, and Jan Storbank Pedersen. *The RAISE Method.* The BCS Practitioner Series. Prentice-Hall, Hemel Hampstead, England, 1995.
19. Carl A. Gunter, Stephen T. Weeks, and Andrew K. Wright. Models and Languages for Digtial Rights. In *Proc. of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)*, pages 4034–4038, Maui, Hawaii, USA, January 2001. IEEE Computer Society Press.
20. C.A. Gunther. *Semantics of Programming Languages.* The MIT Press, Cambridge, Mass., USA, 1992.
21. Joseph Y. Halpern and Vicky Weissman. A Formal Foundation for XrML. In *Proc. of the 17th IEEE Computer Security Foundations Workshop (CSFW'04)*, 2004.
22. Tony Hoare. *Communicating Sequential Processes.* C.A.R. Hoare Series in Computer Science. Prentice-Hall International, 1985.
23. Tony Hoare. Communicating Sequential Processes. Published electronically: `http://www.usingcsp.com/cspbook.pdf`, 2004. Second edition of [22]. See also `http://www.usingcsp.com/`.
24. Markus Holzer, Stefan Katzenbeisser, and Christian Schallhart. Towards a Formal Semantics for ODRL. In *Proc. of the First International ODRL Workshop*, Vienna, Austria, April 2004.

25. R.H. Koenen, J. Lacy, M. Mackay, and S. Mitchell. The long march to interoperable digital rights management. *Proceedings of the IEEE*, 92(6):883–897, June 2004.
26. IPR Systems Pty Ltd. Open Digital Rights Language (ODRL). http://odrl.net, 2001.
27. Gordon E. Lyon. Information Technology: A Quick-Reference List of Organizations and Standards for Digital Rights Management. NIST Special Publication 500-241, National Institute of Standards and Technology, Technology Administration, U.S. Department of Commerce, Oct 2002.
28. S. Michiels, K. Verslype, W. Joosen, and B. De Decker. Towards a Software Architecture for DRM. In *Proceedings of the Fifth ACM Workshop on Digital Rights Management (DRM'05)*, pages 65–74, Alexandria, Virginia, USA, Nov 2005.
29. D. Mulligan and A. Burstein. Implementing copyright limitations in rights expression languages. In *Proc. of 2002 ACM Workshop on Digital Rights Management*, volume 2696 of *Lecture Notes in Computer Science*, pages 137–154. Springer-Verlag, 2002.
30. Deirdre K. Mulligan, John Han, and Aaron J. Burstein. How DRM-Based Content Delivery Systems Disrupt Expectations of "Personal Use". In *Proc. of The 3rd International Workshop on Digital Rights Management*, pages 77–89, Washington DC, USA, Oct 2003. ACM.
31. Riccardo Pucella and Vicky Weissman. A Logic for Reasoning about Digital Rights. In *Proc. of the 15th IEEE Computer Security Foundations Workshop (CSFW'02)*, pages 282–294. IEEE Computer Society Press, 2002.
32. Riccardo Pucella and Vicky Weissman. A Formal Foundation for ODRL. In *Proc. of the Workshop on Issues in the Theory of Security (WIST'04)*, 2004.
33. John C. Reynolds. *The Semantics of Programming Languages*. Cambridge University Press, 1999.
34. A. W. Roscoe. *Theory and Practice of Concurrency*. C.A.R. Hoare Series in Computer Science. Prentice-Hall, 1997. Now available on the net: http://www.comlab.ox.ac.uk/people/bill.roscoe/publications/68b.pdf.
35. Pamela Samuelson. Digital rights management {and, or, vs.} the law. *Communications of ACM*, 46(4):41–45, Apr 2003.
36. David A. Schmidt. *Denotational Semantics: a Methodology for Language Development*. Allyn & Bacon, 1986.
37. Steve Schneider. *Concurrent and Real-time Systems — The CSP Approach*. Worldwide Series in Computer Science. John Wiley & Sons, Ltd., Baffins Lane, Chichester, West Sussex PO19 1UD, England, January 2000.
38. Staff of Merriam Webster. Online Dictionary: `http://www.m-w.com/home.htm`, 2004. Merriam–Webster, Inc., 47 Federal Street, P.O. Box 281, Springfield, MA 01102, USA.
39. Robert Tennent. *The Semantics of Programming Languages*. Prentice–Hall Intl., 1997.
40. G. Winskel. *The Formal Semantics of Programming Languages*. The MIT Press, Cambridge, Mass., USA, 1993.
41. JianWen Xiang and Dines Bjørner. The Electronic Media Industry: A Domain Analysis and a License Language. Technical note, JAIST, School of Information Science, 1-1, Asahidai, Tatsunokuchi, Nomi, Ishikawa, Japan 923-1292, Summer 2006.