Modal Process Logics

Globan 06, DTU, Copenhagen, DK

Luís Caires

Departamento de Informática
Universidade Nova de Lisboa

Road Map

- **Logical Specifications**
  - Operational versus Logical Specifications
  - Logics (review)
  - Program Logics
  - Modal Logics
- **Logics for Concurrency and Distribution**
  - Hennessy-Milner Logics
  - μ-Calculi
  - Spatial Logics
- **Verification Techniques**
  - Model Checking
  - Proof Systems
  - Type Systems

Logical Specifications

Models, Languages, Logics

- **Programming models**
  - Specify computation by means of machines
  - Automata
  - Process Calculi (CCS, π-calculi,...)
  - Abstract and concrete machines: partial functions, JVM, .NET
- **Programming languages**
  - Specify computations by means of programming abstractions
  - Lambda Calculi, Process calculi, rewriting, ...
  - Expressions denote ... values, functions, objects, ...
- **Specification Logics**
  - Specify requirements on machines by means of properties
    - of states, computations, processes
    - of computational artifacts (e.g., networks, messages)
  - What properties are interesting?

“Programming” with Properties

- For design / analysis, one specifies what properties the system or implementation should satisfy
- Properties assert constraints on states, behavior, etc.
- Some properties may be not realizable
  - May be contradictory
  - May be non computable
  - May be not expressible in the intended model
- The meaning of a specification is a property (namely, a set of models).

$$[\![\mathrm{Spec}]\!] = \{ P \mid P \models \mathrm{Spec} \}$$

$$P \in \mathrm{Spec} \textit{ iff } P \models \mathrm{Spec}$$

Specifications

- Target system: nondeterministic two register machine

  Set of States: S is $\mathbb{N} \times \mathbb{N}$ Set of Conditions: C

  Boot state: $s_I \in S$

  Control: Set of conditional rules $R \subseteq S \times S \times C$ ($s \rightarrow s'$ if c)

  Computation Step: $s \rightarrow s'$ if ($s \rightarrow s'$ if c) $\in R$ and $c(s)$

  Computation C: $s_I = s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 \rightarrow$

- SpecA: $\forall C\ \forall s_i \in C$ if $s_i = (x_i, y_i)$ then $x_i + y_i$ is even

- Implementation I1

  $s_I = (0,0)$

  $(x,y) \rightarrow (x+1, y+1)$

- We have I1 $\models$ SpecA (I1 satisfies SpecA)

Specifications

- Target system: nondeterministic two register machine
  - Set of States: S is $\mathbb{N} \times \mathbb{N}$ Set of Conditions: C
  - Boot state: $s_I \in S$
  - Control: Set of conditional rules $R \subseteq S \times S \times C$ ($s \to s'$ if c)
  - Computation Step: $s \to s'$ if ($s \to s'$ if c) $\in R$ and $c(s)$
  - Computation C: $s_I = s_0 \to s_1 \to s_2 \to s_3 \to s_4 \to$
- SpecA: $\forall C\ \forall s_i \in C$ if $s_i = (x_i, y_i)$ then $x_i + y_i$ is even
- Implementation I2
  - $s_I = (1,1)$
  - $(x,y) \to (x+x,\ y+y)$
  - $(x,y) \to (x\text{-}1,\ y+1)$ **if** $(x>0)$
- We also have I2 $\models$ SpecA

Specifications

- Target system: nondeterministic two register machine

  Set of States: S is $\mathbb{N} \times \mathbb{N}$ Set of Conditions: C

  Boot state: $s_I \in S$

  Control: Set of conditional rules $R \subseteq S \times S \times C$ ($s \rightarrow s'$ if c)

  Computation Step: $s \rightarrow s'$ if ($s \rightarrow s'$ if c) $\in R$ and $c(s)$

  Computation C: $s_I = s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 \rightarrow$

- SpecA: $\forall C\ \forall s_i \in C$ if $s_i = (x_i, y_i)$ then $x_i + y_i$ is even
- SpecB: $\exists C\ \exists s_k \in C$ if $s_k = (x_k, y_k)$ then $y_k > x_k$
- Implementation I2

  $s_I = (1,1)$

  $(x,y) \rightarrow (x+x,\ y+y)$

  $(x,y) \rightarrow (x-1,\ y+1)$ (**if** $x>0$)

- So I2 $\vDash$ SpecA; I2 $\vDash$ SpecB; but not I1 $\vDash$ SpecB

"Programming" with Properties

- SpecA: $\forall C\ \forall s_i \in C$ if $s_i = (x_i, y_i)$ then $x_i + y_i$ is even
- SpecB: $\exists C\ \exists s_k \in C$ if $s_k = (x_k, y_k)$ then $y_k > x_k$

- Implementation I1

  $s_I = (0,0)$

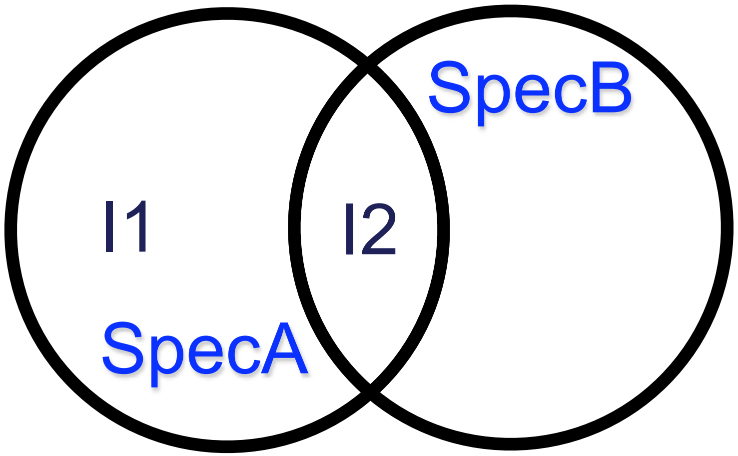
  $(x,y) \rightarrow (x+1, y+1)$

- Implementation I2

  $s_I = (1,1)$

  $(x,y) \rightarrow (x+x,\ y+y)$

  $(x,y) \rightarrow (x-1,\ y+1)$ **(if $x>0$)**



- SpecA $\wedge$ SpecB **refines** SpecA
- SpecC: $\exists C\ \exists s_k \in C\ s_k = (x_k, y_k)$ & $y_k > x_k$ & $x_k + y_k$ is even
- SpecA $\wedge$ SpecB **entails** SpecC

A Comparision

- Operational specifications are naturally monolithic
- An operational specification specifies just one model
- Always realizable by definition
- Good to guide software construction
- Good for analysis

- A logic is a language to express properties (a.k.a. sets of systems / programs)
- Logical specifications are naturally modular
- A logical specification specifies a class of models
  - May be not realizable
  - May be specialized (by refinement)
- Good for design, verification, and analysis
- What logics are there ...?

From Logic to Modal Logic

Propositional Logic

- **Syntax**

  A set $\mathbf{A}$ of atomic propositions (basic properties)

  $$\mathcal{A}, \mathcal{B}, \mathcal{C} ::= \mathrm{a} \in \mathbf{A} \mid \mathcal{A} \wedge \mathcal{B} \mid \neg \mathcal{A} \mid \mathrm{True}$$

- **Semantics**

  The universe: a nonempty set $\mathcal{U}$ of individuals

  A valuation: $\quad v{:}\mathbf{A} \to \wp(\mathcal{U})$

  A formula $\mathcal{A}$ expresses a property $[\![\mathcal{A}]\!]_v \subseteq \mathcal{U}$

  $$[\![\mathrm{a}]\!]_v = v(\mathrm{a})$$

  $$[\![\mathcal{A} \wedge \mathcal{B}]\!]_v = [\![\mathcal{A}]\!]_v \cap [\![\mathcal{B}]\!]_v$$

  $$[\![\mathrm{True}]\!]_v = \mathcal{U}$$

  $$[\![\neg \mathcal{A}]\!]_v = \mathcal{U} \setminus [\![\mathcal{A}]\!]_v$$

Propositional Logic

- Syntax

  A set $\mathbf{A}$ of atomic propositions (basic properties)

  $$\mathcal{A}, \mathcal{B}, \mathcal{C} ::= a \in \mathbf{A} \mid \mathcal{A} \wedge \mathcal{B} \mid \neg\, \mathcal{A} \mid \mathrm{True}$$

- Abreviations ...

  $\mathrm{False} \triangleq \neg\mathrm{True}$

  $\mathcal{A} \vee \mathcal{B} \triangleq \neg\, (\, \neg\mathcal{A} \wedge \neg\mathcal{B}\,)$

  $\mathcal{A} \Rightarrow \mathcal{B} \triangleq \neg\mathcal{A} \vee \mathcal{B}$

- Assertions (judgements; sequents)

  $\mathcal{A}_1, \ldots, \mathcal{A}_n \vdash \mathcal{B}_1, \ldots, \mathcal{B}_m$

- Validity

  $$\mathit{valid}\,(\, \mathcal{A} \vdash \mathcal{B}\,) \triangleq \forall \mathcal{U}.\forall v_{\mathcal{U}}.\quad [\![\wedge_i\, \mathcal{A}_i]\!]_v \subseteq [\![\vee_i\, \mathcal{B}_i]\!]_v$$

  $$\forall \mathcal{U}.\forall v_{\mathcal{U}}.\quad [\![\neg\mathcal{A} \vee \mathcal{B}]\!]_v = \mathcal{U}$$

Propositional Logic

- A Proof System (Sequent Calculus)

$$\mathcal{A} \vdash \mathcal{A}$$

$$\frac{\mathcal{A}, \mathcal{C} \vdash \mathcal{B}}{\mathcal{A} \vdash \neg\mathcal{C}, \mathcal{B}} \qquad \frac{\mathcal{A} \vdash \mathcal{C}, \mathcal{B}}{\mathcal{A}, \neg\mathcal{C} \vdash \mathcal{B}} \qquad \frac{\mathcal{C} \vdash \mathcal{D}, \mathcal{A} \quad \mathcal{C} \vdash \mathcal{D}, \mathcal{A}}{\mathcal{C} \vdash \mathcal{D}, \mathcal{A} \wedge \mathcal{B}}$$

$$\frac{\mathcal{C} \vdash \mathcal{D}}{\mathcal{C}, \mathcal{A} \vdash \mathcal{D}} \qquad \frac{\mathcal{C} \vdash \mathcal{D}}{\mathcal{C} \vdash \mathcal{D}, \mathcal{A}} \qquad \frac{\mathcal{C}, \mathcal{A}, \mathcal{B} \vdash \mathcal{D}}{\mathcal{C}, \mathcal{A} \wedge \mathcal{B} \vdash \mathcal{D}}$$

$$\frac{\mathcal{C}, \mathcal{A} \vdash \mathcal{D}}{\mathcal{C}, \mathcal{A}, \mathcal{A} \vdash \mathcal{D}} \qquad \frac{\mathcal{C} \vdash \mathcal{D}, \mathcal{A}}{\mathcal{C} \vdash \mathcal{D}, \mathcal{A}, \mathcal{A}}$$

- Soundness: if $\mathcal{A} \vdash \mathcal{B}$ then *valid* ($\mathcal{A} \vdash \mathcal{B}$)
- Completeness: if *valid* ($\mathcal{A} \vdash \mathcal{B}$) then $\mathcal{A} \vdash \mathcal{B}$
- Decidability: we can decide $\mathcal{A} \vdash \mathcal{B}$

Predicate Logic

- Syntax

  A set $\mathbf{V}$ of variables (x, y, z)

  A set $\mathbf{P}$ of atomic predicates (p,q,r) (basic relations)

  $$\mathcal{A}, \mathcal{B}, \mathcal{C} ::= \mathrm{p}(x,y) \mid \mathcal{A} \wedge \mathcal{B} \mid \neg\, \mathcal{A} \mid \forall x.\mathcal{A} \mid \mathrm{True}$$

- Semantics

  The universe: a nonempty set $\mathcal{U}$ of individuals

  A interpretation: $\mathcal{I}: \mathbf{P} \to \wp(\mathcal{U} \times \mathcal{U})$

  A valuation: $v: \mathbf{V} \to \mathcal{U}$

- Satisfaction (a model $\mathcal{M}$ satisfies a formula $\mathcal{A}$)

  $$\mathcal{M} \vDash \mathcal{A}$$

- A formula denotes (specifies) a set of models

  $$[\![\mathcal{A}]\!] = \{\, \mathcal{I};v \mid \mathcal{I};v \vDash \mathcal{A} \,\}$$

Predicate Logic

- Semantics

The universe: a nonempty set $\mathcal{U}$ of individuals ($\mathbf{a}$, $\mathbf{b}$)

A interpretation: $\mathcal{I}$: $\mathbf{P} \rightarrow \wp(\mathcal{U} \times \mathcal{U})$

A valuation: v: $\mathbf{V} \rightarrow \mathcal{U}$

A model: $\mathcal{M} = (\ \mathcal{I};v\ )$

- Satisfaction

$\mathcal{I};v \models \mathrm{p}(x,y)$	*iff*	$(\ v(x), v(y)\ ) \in \mathcal{I}(\mathrm{p})$
$\mathcal{M} \models \neg\ \mathcal{A}$	*iff*	*not* $\mathcal{M} \models \mathcal{A}$
$\mathcal{M} \models \mathcal{A} \wedge \mathcal{B}$	*iff*	$\mathcal{M} \models \mathcal{A}$ *and* $\mathcal{M} \models \mathcal{B}$
$\mathcal{I};v \models \forall x.\mathcal{A}$	*iff*	$\forall \mathbf{a} \in \mathcal{U}\ .\ \mathcal{I};v\{x/\mathbf{a}\} \models \mathcal{A}$

- *valid* $(\ \mathcal{A}_1, \ldots, \mathcal{A}_n \vdash \mathcal{B}_1, \ldots, \mathcal{B}_m\ ) \triangleq [\![\wedge_i\ \mathcal{A}_i]\!] \subseteq [\![\vee_i\ \mathcal{B}_i]\!]$

Predicate Logic

- A Proof System

$$\frac{\mathcal{C} \vdash \mathcal{D}, \mathcal{A}}{\mathcal{C} \vdash \mathcal{D}, \forall x.\mathcal{A}} \ (x \textit{ not free in } \mathcal{C},\mathcal{D}) \qquad \frac{\mathcal{C}, \mathcal{A}\{x/y\} \vdash \mathcal{D}}{\mathcal{C}, \forall x.\mathcal{A} \vdash \mathcal{D}}$$

- Soundness: if $\mathcal{A} \vdash \mathcal{B}$ then *valid* ($\mathcal{A} \vdash \mathcal{B}$)
- Completeness: if *valid* ($\mathcal{A} \vdash \mathcal{B}$) then $\mathcal{A} \vdash \mathcal{B}$
- Undecidability: no algorithm can decide $\mathcal{A} \vdash \mathcal{B}$

One can use predicate logic to reason about quite a lot, and certainly about programs, processes, networks,...

But we aim at something more specialized.

The computer science approach: seek and analyze the “right” (preferably tractable) abstractions.

Hoare Logic

- Simple imperative programs

$$P \ ::= \ x := E \mid \text{if } E \text{ then } P \text{ else } P \mid \text{while } E \text{ do } P$$

- Assertions (Hoare triples)

$$\{\mathcal{A}\}\ P\ \{\mathcal{B}\}$$

$\mathcal{A}$, $\mathcal{B}$ are state formulas (predicate logic formulas)

$\mathcal{A}$ is the precondition, assumed to hold **before** P runs

$\mathcal{B}$ is the postcondition, concluded to hold **after** P terminates

- Examples

$\{x = 0\}\ x:=x+1\ \{\ x > 0\ \}$

$\{n \geq 0\}\ x:=1;\ i:=0;$ while $i<n$ do $(i = i+1; x := x * i)$ $\{x = n!\}$

N.B. Names in formulas (x,i,n) refer to program variables

Hoare Logic

- A set of program variables $\mathcal{V}$
- A state is a valuation s: $\mathcal{V} \rightarrow$ `int`
- Semantics of programs

Expression evaluation: $[\![-]\!]_s$ maps E to $[\![E]\!]_s$

$[\![n]\!]_s = n$; $[\![x]\!]_s = s(x)$; $[\![E + F]\!]_s = [\![E]\!]_s + [\![F]\!]_s$; etc ...

Transition relation: go from s to r by running P: $\quad s \xrightarrow{P} r$

$$s \xrightarrow{x:=E} s\{x / [\![E]\!]_s\}$$

$$\frac{[\![E]\!]_s = \text{true} \quad s \xrightarrow{P} r \quad r \xrightarrow{\text{while } E \text{ do } P} p}{s \xrightarrow{\text{while } E \text{ do } P} p} \qquad \frac{s \xrightarrow{P} r \quad r \xrightarrow{Q} p}{s \xrightarrow{P;Q} p}$$

Hoare Logic

Semantics of assertions

valid ($\{\mathcal{A}\}\ P\ \{\mathcal{B}\}$) $\triangleq$

$$\forall s.\ \text{if } s \vDash \mathcal{A} \ \text{ and } s \xrightarrow{P} r \text{ then } r \vDash \mathcal{B}$$

Proof System

Some proof rules such as

$$\frac{\{\mathcal{A} \wedge E\}\ P\ \{\mathcal{A}\}}{\{\mathcal{A}\}\ \text{while } E \text{ do } P\ \{\mathcal{A} \wedge \neg E\}} \qquad \frac{\mathcal{A}\ \mathcal{A}' \quad \{\mathcal{A}'\}\ P\ \{\mathcal{B}'\} \quad \mathcal{B}'\ \mathcal{B}}{\{\mathcal{A}\}\ P\ \{\mathcal{B}\}}$$

Useful to write safety specifications

Do not ensure the program will actually do something:

If P does not terminate, then $\{\mathcal{A}\}\ P\ \{\mathcal{B}\}$ is valid for any $\mathcal{A},\mathcal{B}$

But if something happens, all will be ok (if the spec is)

Modal Logic as Program Logic

- Modal logics talk about structures consisting of many suitably related states (or "worlds")

  Each world is a "classical" model (*e.g.*, a boolean algebra).

  $$e.g.,\ s \models \mathcal{A}$$

  The novelty: special operators (called modalities) allowing us to "jump" from world to world, or quantify over worlds.

  $$e.g.,\ and\ s \longrightarrow r\ and\ r \models \mathcal{A}\ then\ s \models [\mathrm{next}]\mathcal{A}$$

- Specific modal logics may talk about time, behavior, space, resources, data, knowledge, necessity, etc...
- "Program Logic" as a modal logic:

  $$\mathcal{A}, \mathcal{B}, \mathcal{C} ::= \mathrm{p}(x,y) \mid \mathcal{A} \wedge \mathcal{B} \mid \neg\, \mathcal{A} \mid \forall x.\mathcal{A} \mid [P]\mathcal{A}$$

  $$s \models [P]\mathcal{A} \quad iff \quad \forall r.\ if\ s \xrightarrow{P} r\ then\ r \models \mathcal{A}$$

Modal Logic (Classical)

- Worlds

  Intuition: a "world" is a state s (a boolean valuation)

- Accessibility (relation between worlds)

  A transition relation $s \longrightarrow r$

  Intuition: for each world s, there are some "alternative" worlds, namely those worlds r such that $s \longrightarrow r$

- Syntax

$$\mathcal{A}, \mathcal{B}, \mathcal{C} ::= \mathrm{a} \in \mathbf{A} \mid \mathcal{A} \wedge \mathcal{B} \mid \neg \mathcal{A} \mid \mathrm{True} \mid \Box \mathcal{A}$$

- Models $\mathcal{M} = (v, \rightarrow)$

  A valuation $v{:}\mathbf{A} \rightarrow \wp(\mathcal{S})$

  - Says what holds at each world

  A transition system $\rightarrow$

  - Says what worlds are compatible / nearby / next / ...

Modal Logic (Classical)

- Model $\mathcal{M}$

  A valuation v:$\mathbf{A} \longrightarrow \wp(\mathcal{S})$

  A transition system $\longrightarrow$

- Semantics

  $s \vDash$ a *iff* s $\in v$(a)

  $s \vDash \Box \mathcal{A}$ *iff* $\forall r$. *if* $s \longrightarrow r$ then $r \vDash \mathcal{A}$

  valid ($\mathcal{A}_1, ..., \mathcal{A}_n \vdash \mathcal{B}_1, ..., \mathcal{B}_m$) $\triangleq [\![\wedge_i \mathcal{A}_i]\!] \subseteq [\![\vee_i \mathcal{B}_i]\!]$

- Examples of (abstract) modal reasoning

  valid ($\mathcal{A}$) implies valid ($\Box \mathcal{A}$) (Necessitation)

  $\Box(\mathcal{A} \Rightarrow \mathcal{B}) \Rightarrow (\Box \mathcal{A} \Rightarrow \Box \mathcal{B})$ (Axiom K)

  $\Box \mathcal{A} \Rightarrow \mathcal{A}$ (Axiom T; is this sensible?)

- What are the axioms? (please specify your model ...)

More Modal Logics

- **Linear Time Temporal Logic**
  - Models are a time line
  - Amir Pnueli proposed LTL (1977) for reasoning about concurrent and non terminating programs such as operating systems.
- **Branching Time Temporal Logic**
  - Models are trees, each instant may have different futures
  - Useful to reason about non-determinism
- **Computational Tree Logic**
  - Models are trees
  - CTL distinguishes between “path” and “state” modalities
- **Process Logics**
  - Hennessy-Milner Logics
  - μ-Calculus
  - Spatial Logics

Hennessy-Milner Logics

Hennessy-Milner Logic

- A modal logic for labeled transition systems

  Labeled transition system

  A set $\mathcal{A}$ of actions (α,β)

  A set $\mathcal{S}$ of states

  A labeled transition relation $\mathcal{T} \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{S}$

  N.B. Write $s \xrightarrow{\alpha} r$ when $(s, \alpha, r) \in \mathcal{T}$

- Syntax

$$\mathcal{A}, \mathcal{B}, \mathcal{C} ::= \wedge_{i\in I} \mathcal{A}_i \mid \neg \mathcal{A} \mid \langle\alpha\rangle\mathcal{A}$$

- Remarks

  Infinitary syntax: True $\triangleq \wedge_{i\in\varnothing} \mathcal{A}_i$

  No propositional symbols; the logic just observes actions

  Hint: **HML** is a modal logic of pure behavior

Hennessy-Milner Logic

- A modal logic for labeled transition systems

  Labeled transition system

  A set $\mathcal{A}$ of actions (α,β)

  A set $\mathcal{S}$ of states

  A labeled transition relation $\mathcal{T} \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{S}$

  N.B. Write $s \xrightarrow{\alpha} r$ when $(s, \alpha, r) \in \mathcal{T}$

- Syntax (finitary version)

$$\mathcal{A}, \mathcal{B}, \mathcal{C} ::= \mathcal{A} \wedge \mathcal{B} \mid \neg \mathcal{A} \mid \langle\alpha\rangle\mathcal{A}$$

- Modalities $\langle\alpha\rangle\mathcal{A}$ observe nearby states

  In some α-next state $\mathcal{A}$ holds: $\langle\alpha\rangle\mathcal{A}$

  In some α-next states $\mathcal{A}$ holds: $[\alpha]\mathcal{A} \triangleq \neg\langle\alpha\rangle\neg\mathcal{A}$

Hennessy-Milner Logic

- A modal logic for labeled transition systems

  Labeled transition system

  A set $\mathcal{A}$ of actions (α,β)

  A set $\mathcal{S}$ of states

  A labeled transition relation $\mathcal{T} \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{S}$

  N.B. Write $s \xrightarrow{\alpha} r$ when $(s, \alpha, r) \in \mathcal{T}$

- Satisfaction

$s \vDash \langle\alpha\rangle\mathfrak{A}$	*iff*	*exists* $r.\ s \xrightarrow{\alpha} r$ *and* $r \vDash \mathfrak{A}$
$s \vDash \neg\, \mathfrak{A}$	*iff*	*not* $s \vDash \mathfrak{A}$
$s \vDash \mathfrak{A} \wedge \mathfrak{B}$	*iff*	$s \vDash \mathfrak{A}$ *and* $s \vDash \mathfrak{B}$

Hennessy-Milner Logic (CCS)

- Calculus of Communicating Systems (Milner)

  A set $\mathcal{A}$ of actions (α, τ) where $\alpha = n$ or $\alpha = \overline{n}$

  A set $\mathcal{P}$ of processes

$$P, Q, R ::= \mathbf{0} \mid P \mid Q \mid \alpha.P \mid (\mathbf{new}\ n)P \mid x \mid \mathbf{rec}\ x.P$$

- Labeled Transition System for CCS

$$\alpha.P \xrightarrow{\alpha} P \qquad \frac{P \xrightarrow{\alpha} Q \quad P' \xrightarrow{\overline{\alpha}} Q'}{P \mid P' \xrightarrow{\tau} Q \mid Q'} \qquad \frac{P \xrightarrow{\alpha} Q}{P \mid R \xrightarrow{\alpha} Q \mid R}$$

$$\frac{P \xrightarrow{\alpha} Q}{(\mathbf{new}\ n)P \xrightarrow{\alpha} (\mathbf{new}\ n)Q}\ (n \text{ not in } \alpha) \qquad \frac{P\{x / \mathbf{rec}\ x.P\} \xrightarrow{\alpha} Q}{\mathbf{rec}\ x.P \xrightarrow{\alpha} Q}$$

- We interpret Hennessy-Milner Logic on CCS

Hennessy-Milner Logic (CCS)

- $P \models \langle \bar{n} \rangle \mathrm{True}$

  P can perform an output on n
- $P \models [n]\mathrm{False}$

  P refuses to perform an input on n
- $P \models \langle n \rangle [n]\mathrm{False}$

  P can input on n, and then refuse to perform an input on n
- $P \models \langle n \rangle \mathrm{True} \wedge [n][\tau]\mathrm{False}$

  P can input on n, but after any such input will get stuck
- $P \models \langle n \rangle \langle m \rangle \mathrm{True} \wedge \langle n \rangle [m]\mathrm{False}$

  P can do n and then do m, but also do n and after refuse m
- $P \models [n]\langle m \rangle \mathrm{True} \wedge [n][m]\mathrm{False}$

  P can do m after any n, but also refuse m after any n

  So, P cannot really do n

Hennessy-Milner Logic

- Other useful modalities

  All definable in the infinitary logic (how?)

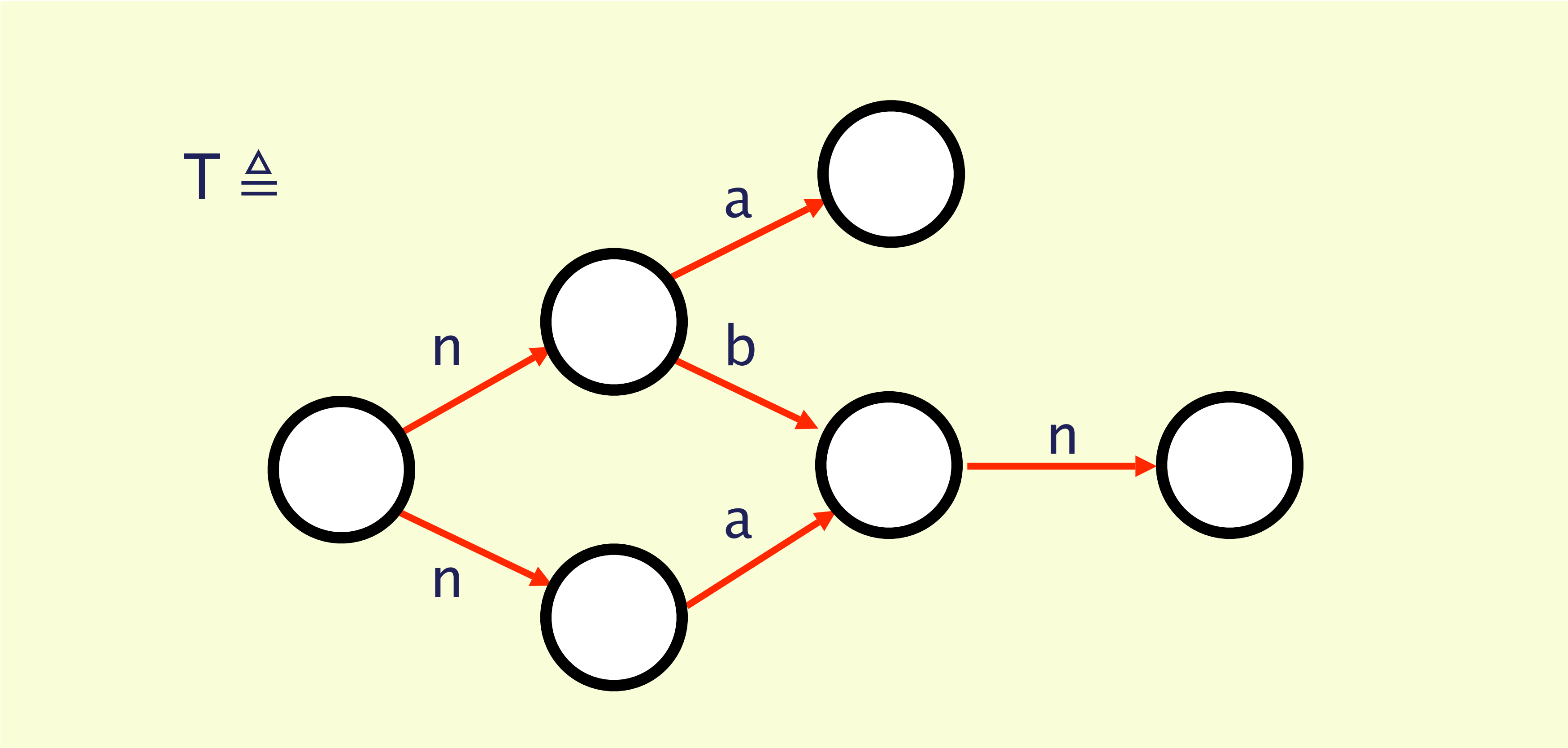
$$\mathcal{A}, \mathcal{B}, \mathcal{C} ::= \mathcal{A} \wedge \mathcal{B} \mid \neg\, \mathcal{A} \mid \langle S \rangle \mathcal{A} \mid \langle * \rangle \mathcal{A} \mid \langle - S \rangle \mathcal{A}$$

- Satisfaction

$s \vDash \langle S \rangle \mathcal{A}$	*iff*	*exists* r, $\alpha \in S$ *and* $s \xrightarrow{\alpha} r$ *and* $r \vDash \mathcal{A}$
$s \vDash \langle * \rangle \mathcal{A}$	*iff*	*exists* r, β. $s \xrightarrow{\beta} r$ *and* $r \vDash \mathcal{A}$
$s \vDash \langle - S \rangle \mathcal{A}$	*iff*	*exists* r, $\alpha \notin S$ *and* $s \xrightarrow{\alpha} r$ *and* $r \vDash \mathcal{A}$
$s \vDash \neg\, \mathcal{A}$	*iff*	*not* $s \vDash \mathcal{A}$
$s \vDash \mathcal{A} \wedge \mathcal{B}$	*iff*	$s \vDash \mathcal{A}$ *and* $s \vDash \mathcal{B}$

Characterize the Behavior

- Find a set of HML formulas that precisely characterize the labeled transition system T below:



- The specification thus obtained is usually called a "characteristic formula" of T

What can we say about?

- Express in HML some properties of the processes

$P1 \triangleq \overline{\mathsf{n}}.(\mathsf{n}.\mathbf{0} \mid \overline{\mathsf{n}}.\mathbf{0})$

$P2 \triangleq \textbf{rec}\ x.\ \mathsf{n}.(\overline{\mathsf{n}}.\mathbf{0} \mid x)$

$P3 \triangleq (\textbf{new}\ n)\,(P1 \mid P2)$

Can we find a model of?

- Try to find CCS models for the HML formulas below

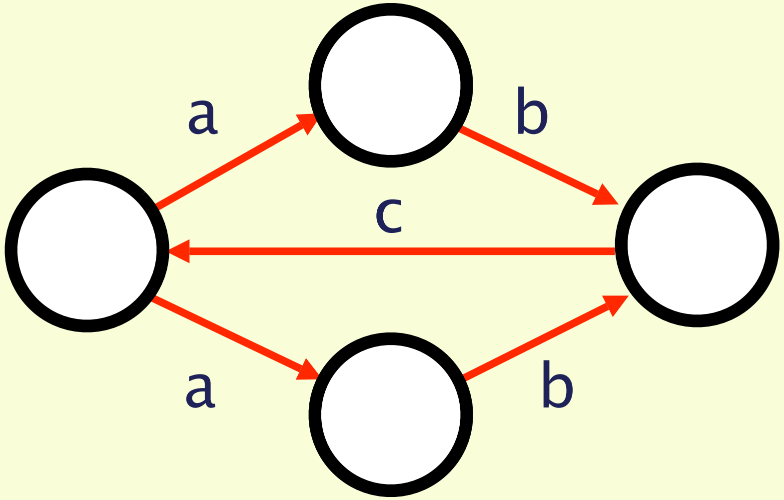
$\mathcal{F} \triangleq \langle n \rangle \langle m \rangle \mathrm{True} \wedge \langle n \rangle [m] \mathrm{False}$

$\mathcal{G} \triangleq \langle n \rangle \langle \overline{n} \rangle \mathrm{True} \wedge [\tau] [*] \mathrm{False}$

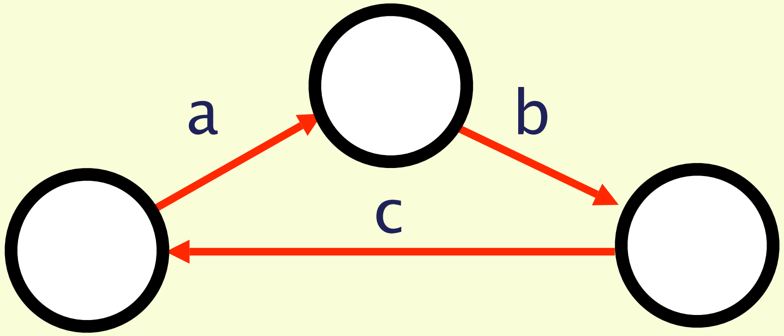
$\mathcal{H} \triangleq [-\{\overline{n}, \tau\}] \mathrm{False} \wedge \langle \tau \rangle \langle n \rangle \mathrm{True}$

Can we distinguish by HML properties ?

$T1 \triangleq$

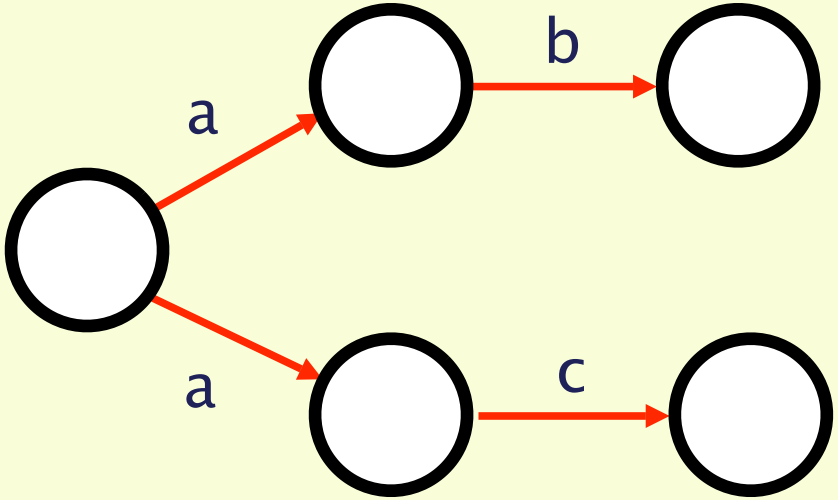


$T2 \triangleq$

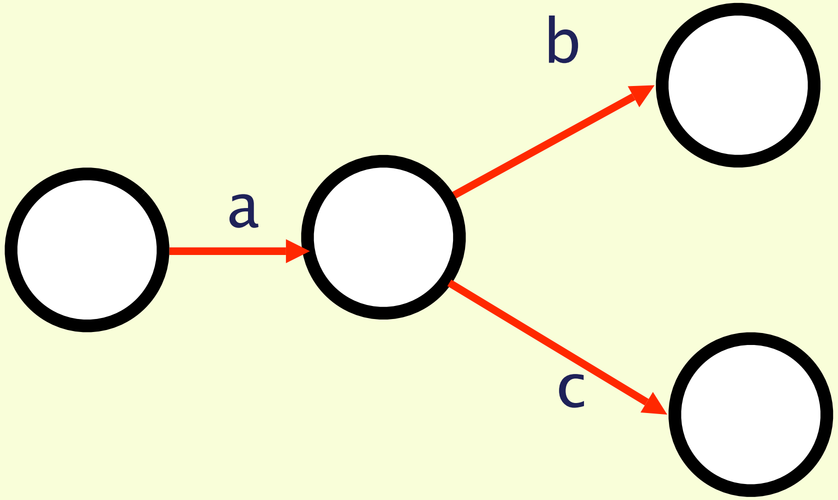


Can we distinguish by HML properties ?

$T3 \triangleq$



$T4 \triangleq$



Separation and Expressiveness

- Indistinguishability in a general modal logic **L**

  States s and r are indistinguishable in **L** if

  $$\forall \mathcal{A}.\ s \models \mathcal{A} \text{ iff if } r \models \mathcal{A}$$

- We define logical equivalence of states, noted $=_L$, by

  $$s =_L r \quad \triangleq \quad \forall \mathcal{A}.\ s \models \mathcal{A} \text{ iff if } r \models \mathcal{A}$$

- Logical equivalence is an equivalence relation on $\mathcal{S}$
- A logic **L** is finer (has more separation power) than a logic **L'** if $=_L \subseteq\ =_{L'}$
- Given a property $\mathcal{P} \subseteq \mathcal{S}$, we say $\mathcal{P}$ is expressible in **L** if there is a formula $\mathcal{A}$ of **L** such that $[\![\mathcal{A}]\!] = \mathcal{P}$
- A logic **L** is more expressive than a logic **L'** if every property expressive in **L'** may be also expressed in **L**

Bisimulation

- Caracterizes coinductively indistinguishable states

  A binary relation $\mathcal{B} \subseteq \mathcal{S} \times \mathcal{S}$ is a bisimulation if

  for all $(s,r) \in \mathcal{B}$

  if $s \xrightarrow{\alpha} s'$ then there is r' such that $r \xrightarrow{\alpha} r'$ and $(s',r') \in \mathcal{B}$

  if $r \xrightarrow{\alpha} r'$ then there is s' such that $s \xrightarrow{\alpha} s'$ and $(s',r') \in \mathcal{B}$
- Bisimulations are equivalence relations
- Bisimulations are closed under arbitrary unions
- Bisimilarity ~ is the greatest bisimulation

$$\sim \; = \bigcup \{ \mathcal{B} \mid \mathcal{B} \text{ is a bisimulation } \}$$

- We may define similar notions for most modal models (Kripke models); *e.g.*, we may also want to observe state valuations, etc, ...

Separation Power of HML

- Theorem

$$\text{If } P \sim Q \text{ then } P =_L Q$$

This follows from

- Lemma

For any formula $\mathcal{A}$, if $P \models \mathcal{A}$ and $P \sim Q$ then $Q \models \mathcal{A}$

Proof: induction on the structure of $\mathcal{A}$.

HML does not distinguish between bisimilar states

N.B. Analogous results should hold for most modal logics, given a suitable notion of bisimulation.

HML is extensional with relation to pure behaviors.

Some interesting properties of processes are not extensional (deadlock, stuckness, race freeness).

Separation Power of HML

- Theorem

$$\text{If } P =_L Q \text{ then } P \sim Q$$

This follows from

- Lemma

$$=_L \text{ is a strong bisimulation}$$

Proof: it may be more easy to show the converse:

$$\text{If } P \nsim Q \text{ then } P \neq_L Q$$

- HML can only observe processes up to a finite depth

If $P \models \mathcal{A}$ and $P \sim_k Q$ then $Q \models \mathcal{A}$ for $k \geq \text{size}(\mathcal{A})$

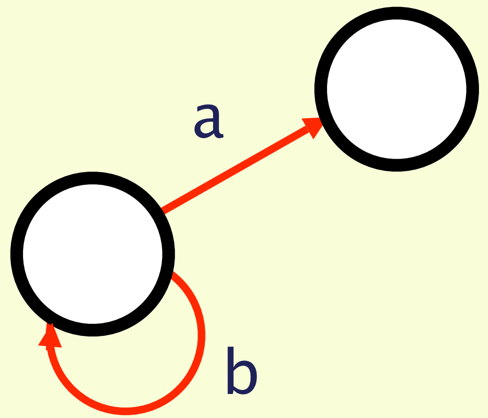
So, we may have $P \sim_k Q$ and $P \nsim Q$.

N.B. If $P \nsim Q$ then there is k such that $P \nsim_k Q$.

The μ-Calculus

Specifying an infinite amount of information

$L \triangleq$



dead $\triangleq$ [*]False

$\langle a \rangle \text{dead} \wedge \langle b \rangle \text{True}$

$\langle a \rangle \text{dead} \wedge \langle b \rangle (\langle a \rangle \text{dead} \wedge \langle b \rangle \text{True})$

$\langle a \rangle \text{dead} \wedge \langle b \rangle (\langle a \rangle \text{dead} \wedge \langle b \rangle (\langle a \rangle \text{dead} \wedge \langle b \rangle \ldots))$

- $\nu X.(\langle a \rangle \text{dead} \wedge \langle b \rangle X)$

The μ-Calculus (Kozen)

- Syntax (extension of HML with fixed points operator)

  A set $\mathcal{V}$ of propositional variables (x, y, z)

  $$\mathcal{A}, \mathcal{B}, \mathcal{C} ::= \mathcal{A} \wedge \mathcal{B} \mid \neg\mathcal{A} \mid \langle\alpha\rangle\mathcal{A} \mid \nu X.\mathcal{A} \mid X \ (\in \mathcal{V})$$

- Satisfaction

  A valuation: $v: \mathcal{V} \to \wp(\mathcal{S})$

$s \models_v \langle\alpha\rangle\mathcal{A}$	*iff*	*exists* $r.\ s \xrightarrow{\alpha} r$ *and* $r \models_v \mathcal{A}$
$s \models_v \neg\mathcal{A}$	*iff*	*not* $s \models_v \mathcal{A}$
$s \models_v \mathcal{A} \wedge \mathcal{B}$	*iff*	$s \models_v \mathcal{A}$ *and* $s \models_v \mathcal{B}$
$s \models_v X$	*iff*	$s \in v(X)$
$s \models_v \nu X.\mathcal{A}$	*iff*	$s \in \mathrm{gfp}(\lambda\mathcal{P}.\ [\![\mathcal{A}]\!]_{v[X/\mathcal{P}]})$
$[\![\nu X.\mathcal{A}]\!]_v$	=	$\bigcup\{\mathcal{P} \subseteq \mathcal{S} \mid \mathcal{P} \subseteq [\![\mathcal{A}]\!]_{v[X/\mathcal{P}]}\}$

The μ-Calculus

- Least fixed point

  $\mu \mathcal{X}.\mathcal{A} \triangleq \neg\nu\mathcal{X}.\neg\mathcal{A}\{\mathcal{X}/\neg\mathcal{X}\}$

- Always $\mathcal{A}$ (under the actions in S)

  inv $\mathcal{A} \triangleq \nu\mathcal{X}.(\mathcal{A} \wedge [S]\mathcal{X})$

  useful to specify invariant properties of systems

- Possibly $\mathcal{A}$ (after some actions in S)

  poss $\mathcal{A} \triangleq \mu\mathcal{X}.(\mathcal{A} \wedge \langle S\rangle\mathcal{X})$

  N.B. **poss** $\mathcal{A}$ = ¬ **inv** $\neg\mathcal{A}$

- $\mathcal{A}$ **until** $\mathcal{B}$ (under the actions in S)

  $\mathcal{A}$ **until** $\mathcal{B} \triangleq \nu\mathcal{X}.(\mathcal{B} \vee (\mathcal{A} \wedge [S]\mathcal{X}))$

The μ-Calculus

- Eventually $\mathcal{A}$ (after some actions in S)

  ev $\mathcal{A} \triangleq \mu \mathcal{X}.(\mathcal{A} \wedge \langle S \rangle \mathrm{True} \wedge [S]\mathcal{X})$

- $\mathcal{A}$ **suntil** $\mathcal{B}$ (under the actions in S)

  $\mathcal{A}$ **suntil** $\mathcal{B} \triangleq \nu \mathcal{X}.(\mathcal{B} \vee (\mathcal{A} \wedge \langle S \rangle \mathrm{True} \wedge [S]\mathcal{X}))$

- A process is insistent for action s if in every infinite computation sequence, s is executed infinitely often.

  insistent s $\triangleq$

- A process is unfair w.r.t. the action s if it may always perform s, but in some possible infinite computation sequence it never actually gets to perform s.

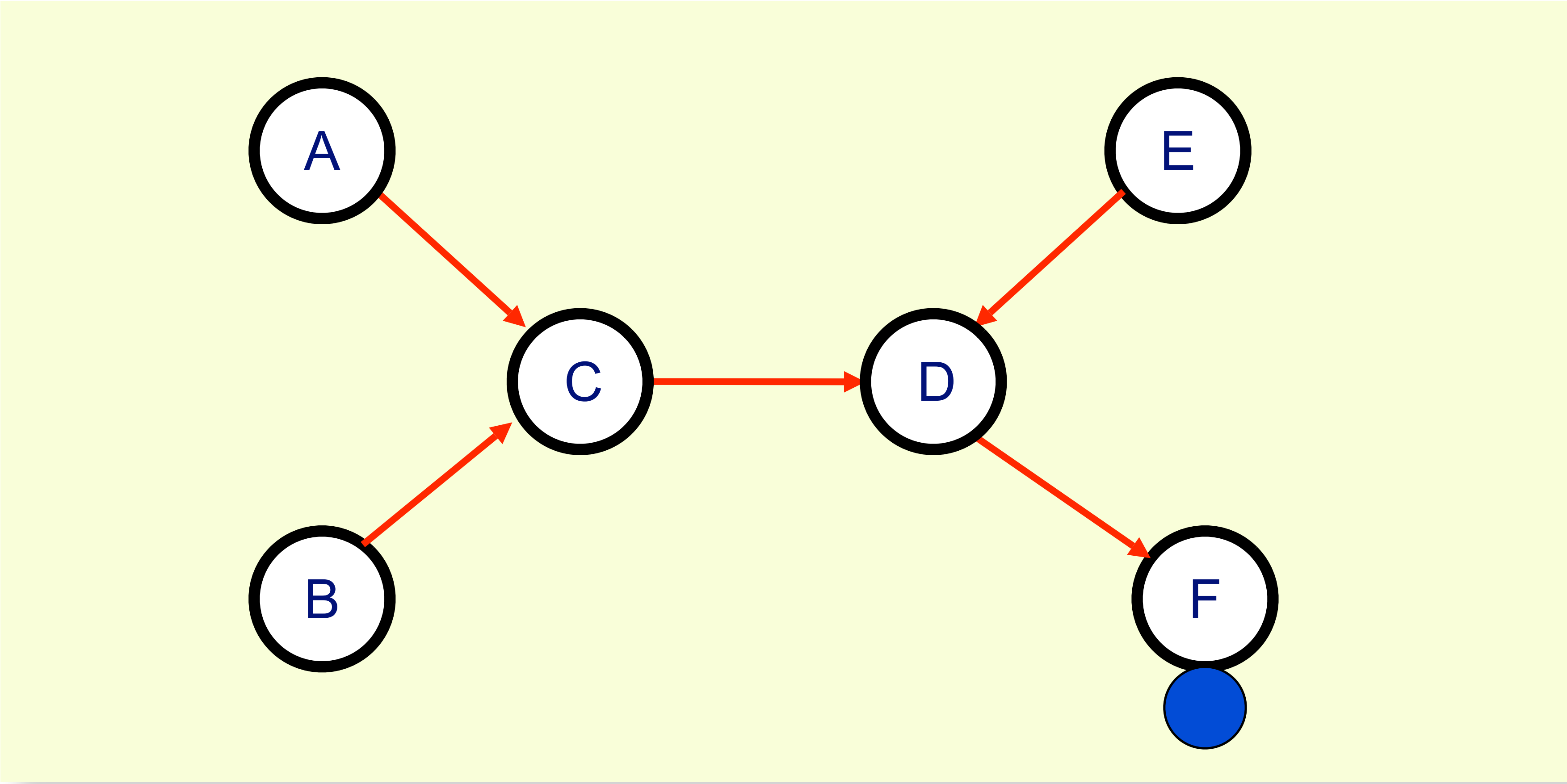
  unfair s $\triangleq$

Spatial Logics for Concurrency

Reasoning about Distributed Systems

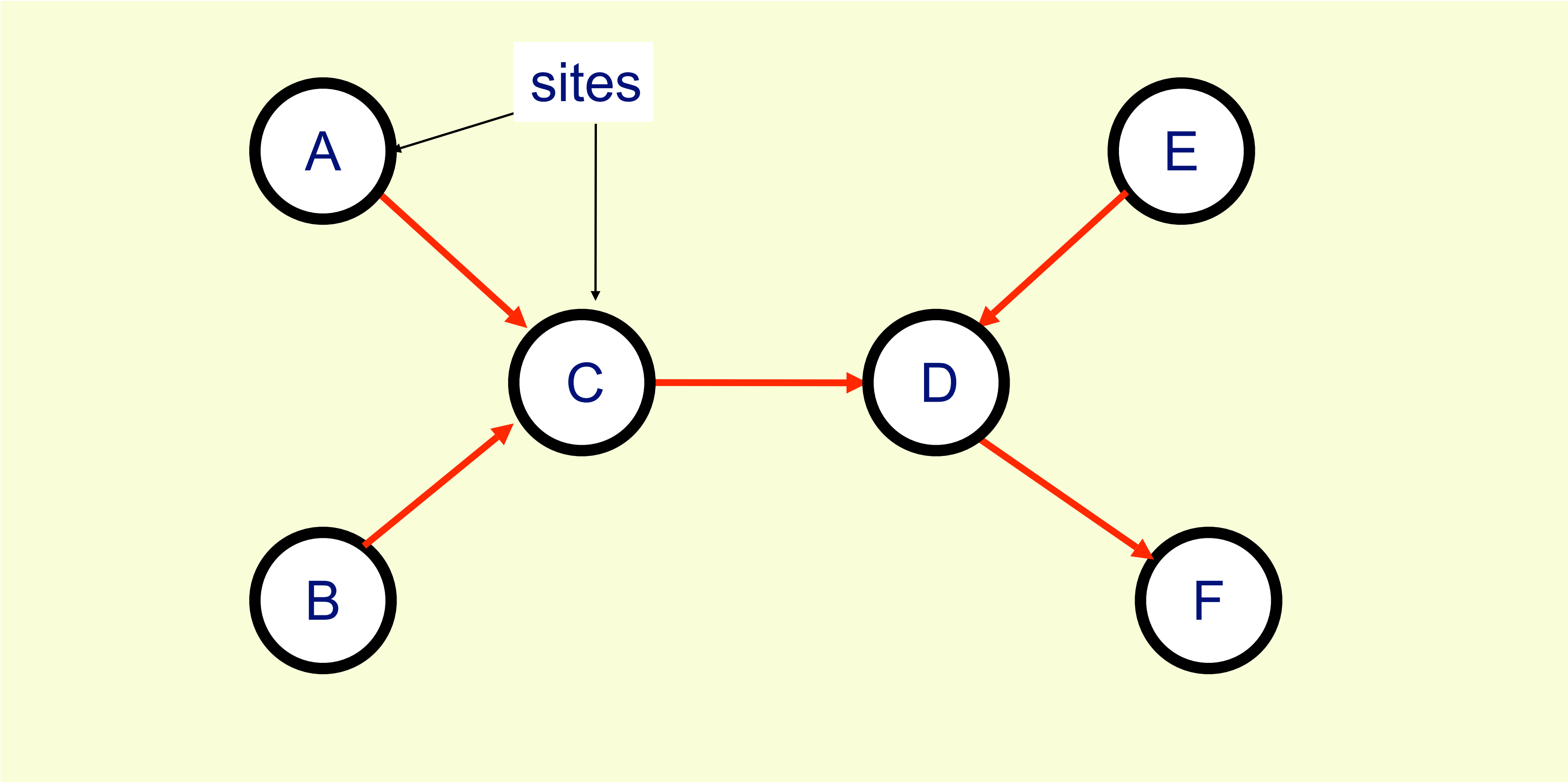
- **Traditional focus**
  - Abstract from irrelevant implementation details
  - Study extensional models of *processes as pure behaviors*
- **A focus on Distributed Systems**
  - Systems where behavior is *spatially* distributed among *sites*
  - Processes behave in time, but site and move in space
  - Structure of space may change during computation
  - Non-behavioral aspects just cannot be abstracted way
    - *E.g.*, geometry, topology, identity, naming, …
  - Several kinds of spatial structure …
  - Several possibilities for space / behaviour interaction …
- **Operational Techniques**
  - Spatial properties also useful for compositional reasoning
  - Spatial logics can also provide a basis for type systems

Example: Resource Discovery



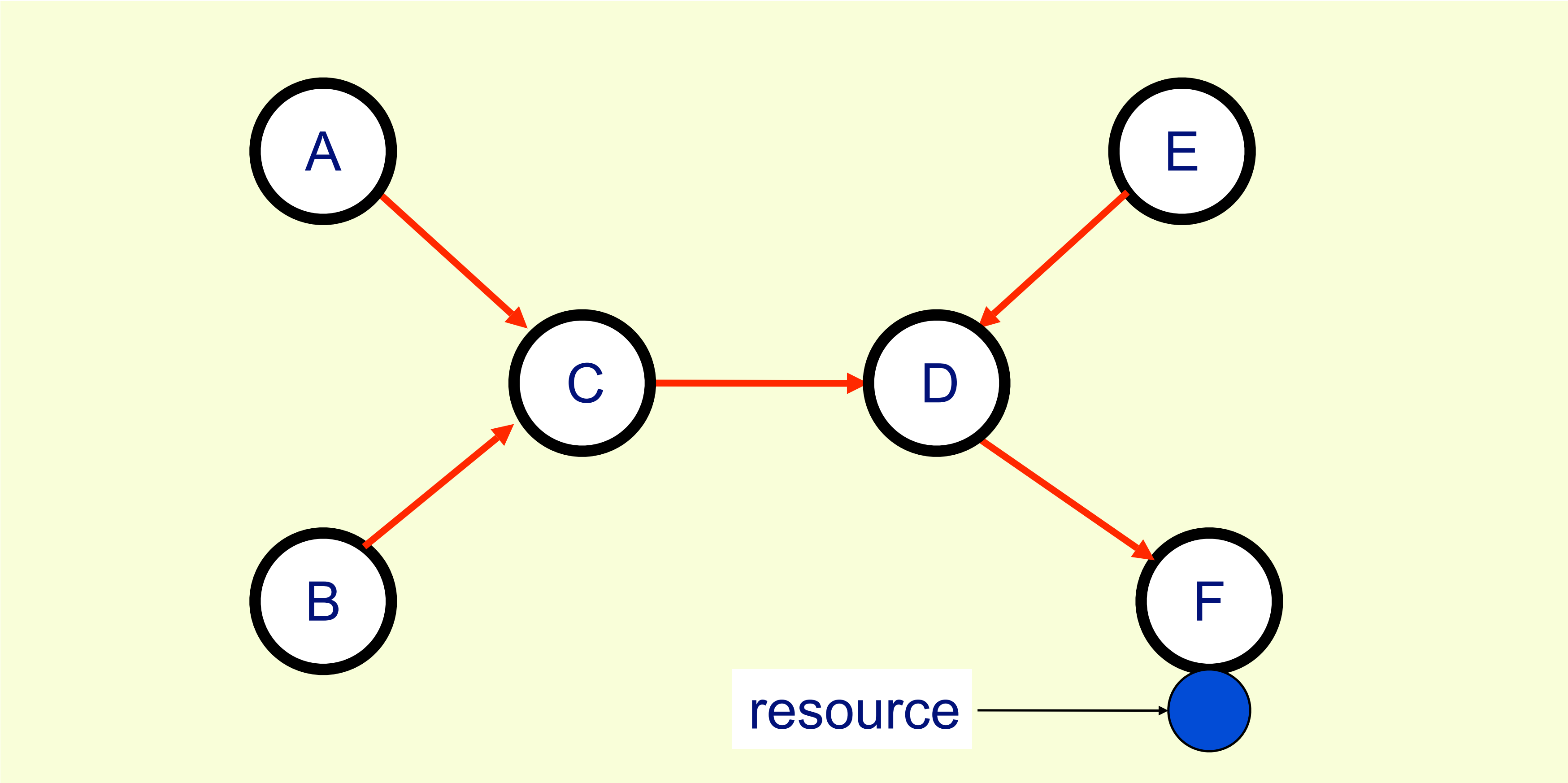
A Distributed Directory Protocol

Example: Resource Discovery



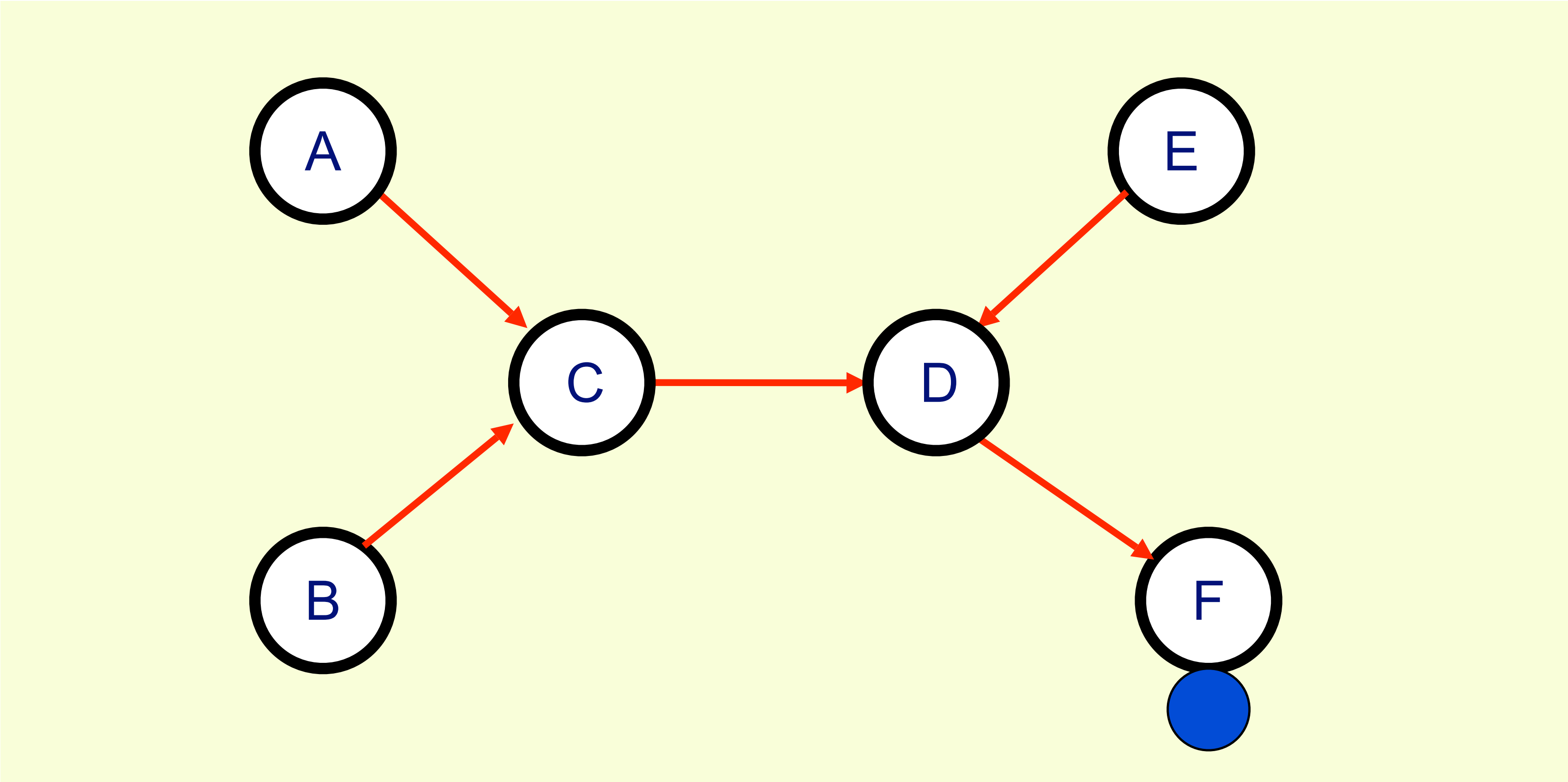
A Distributed Directory Protocol

Example: Resource Discovery



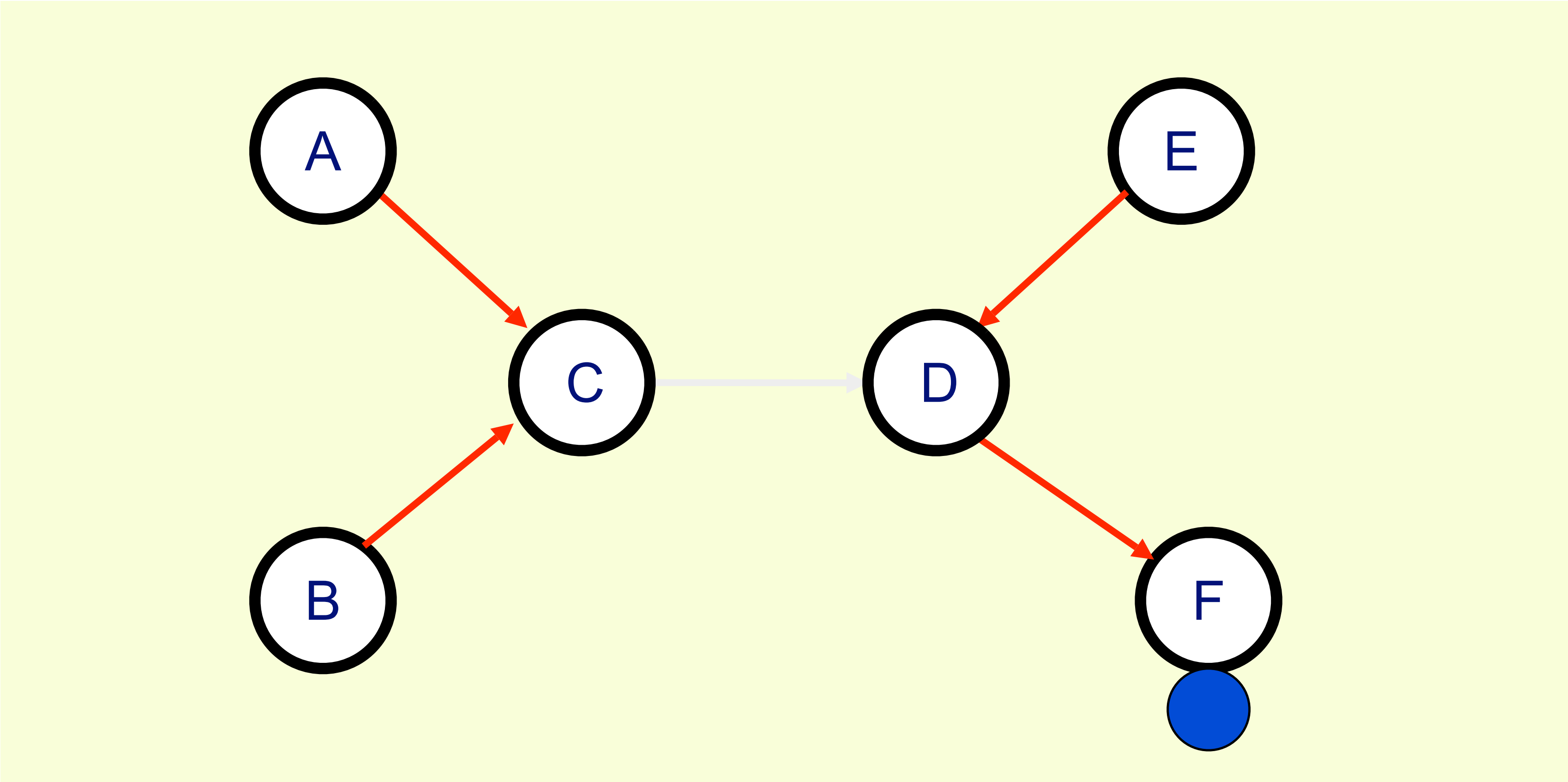
A Distributed Directory Protocol

Spatial Properties



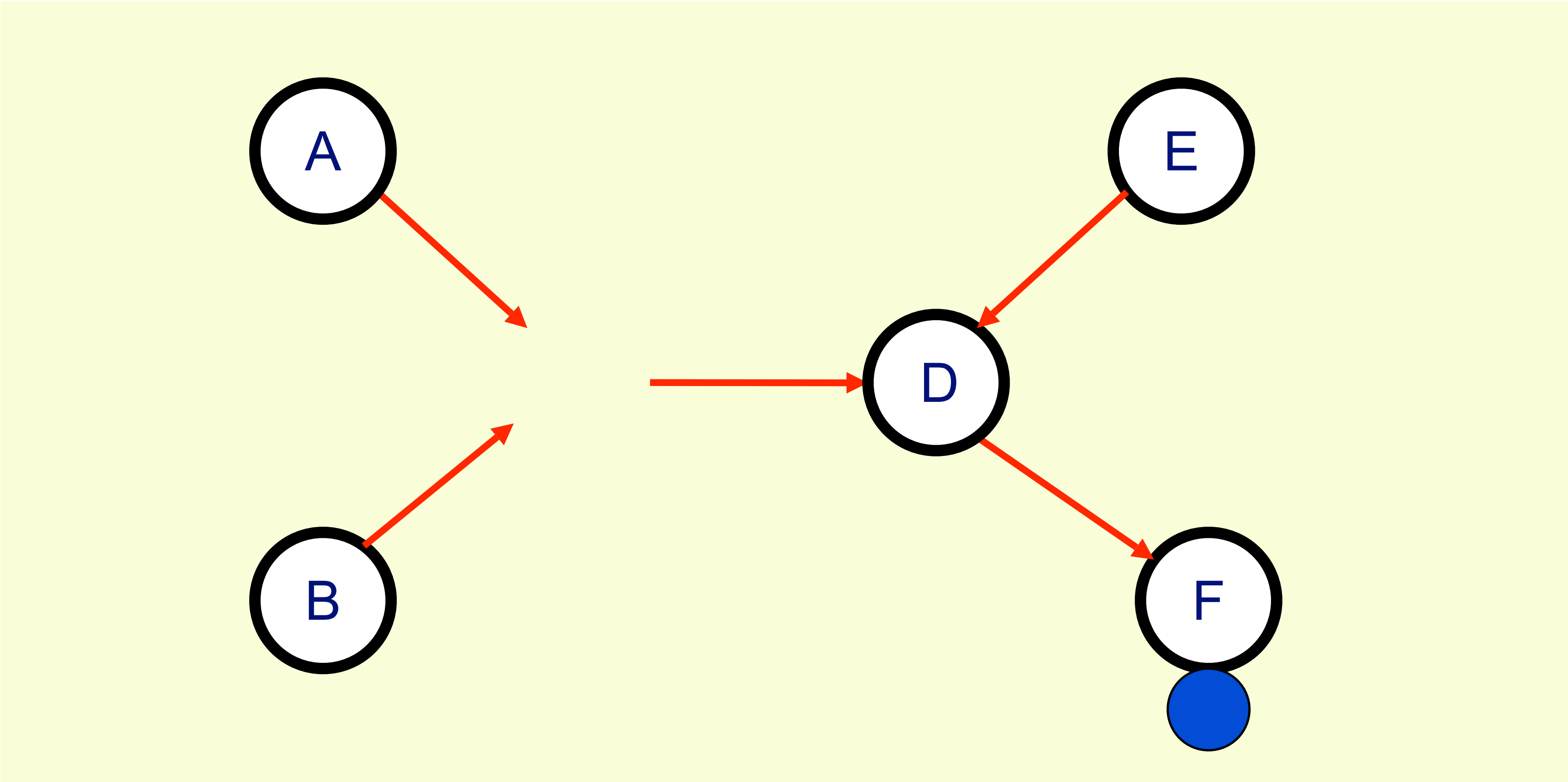
- "The connection structure is a spanning tree"
- N.B.: Key for proving correctness of the protocol.

Spatial Properties



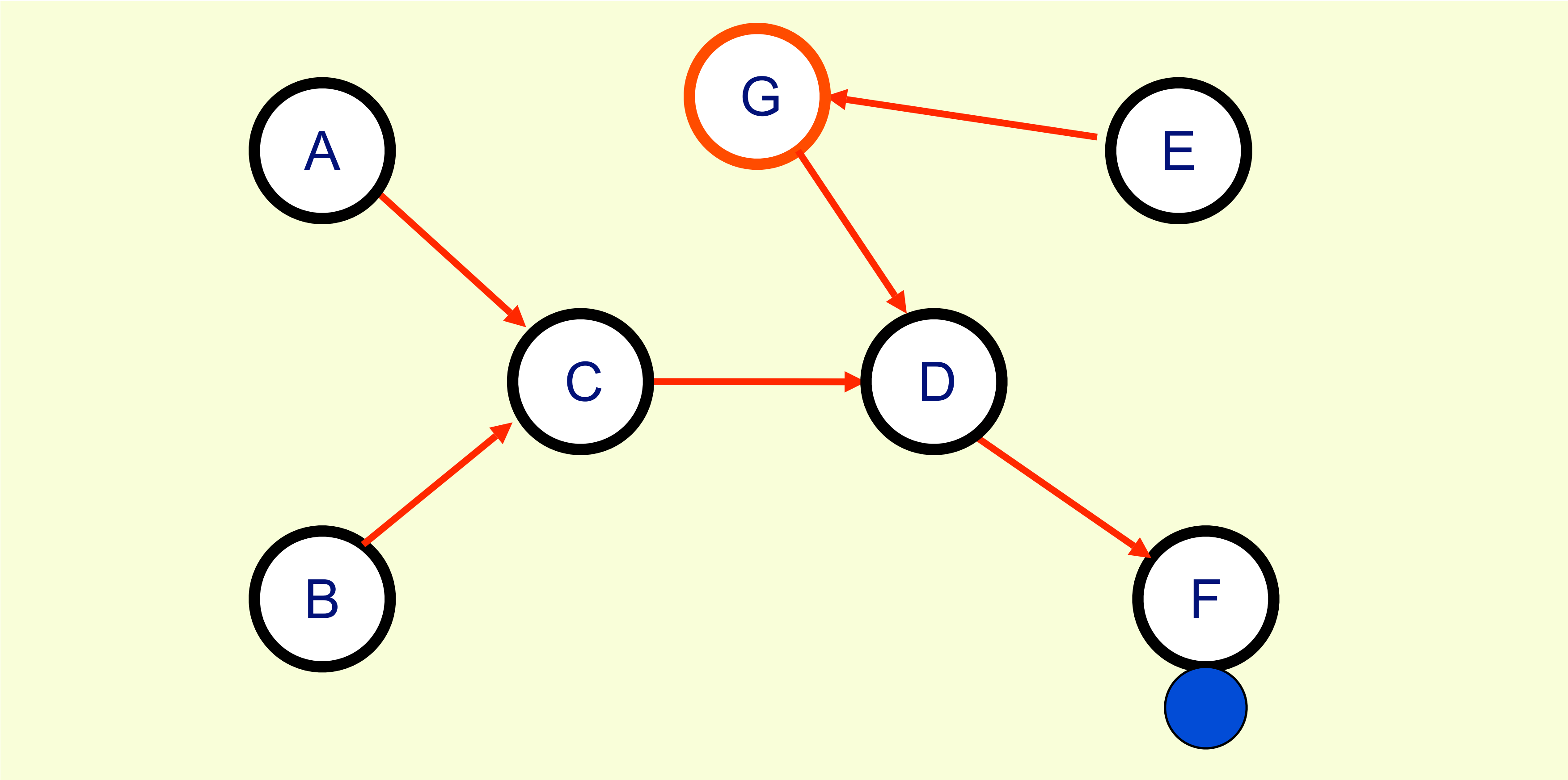
- “Some link failed; the network is partitioned”

Spatial - Behavioral Properties



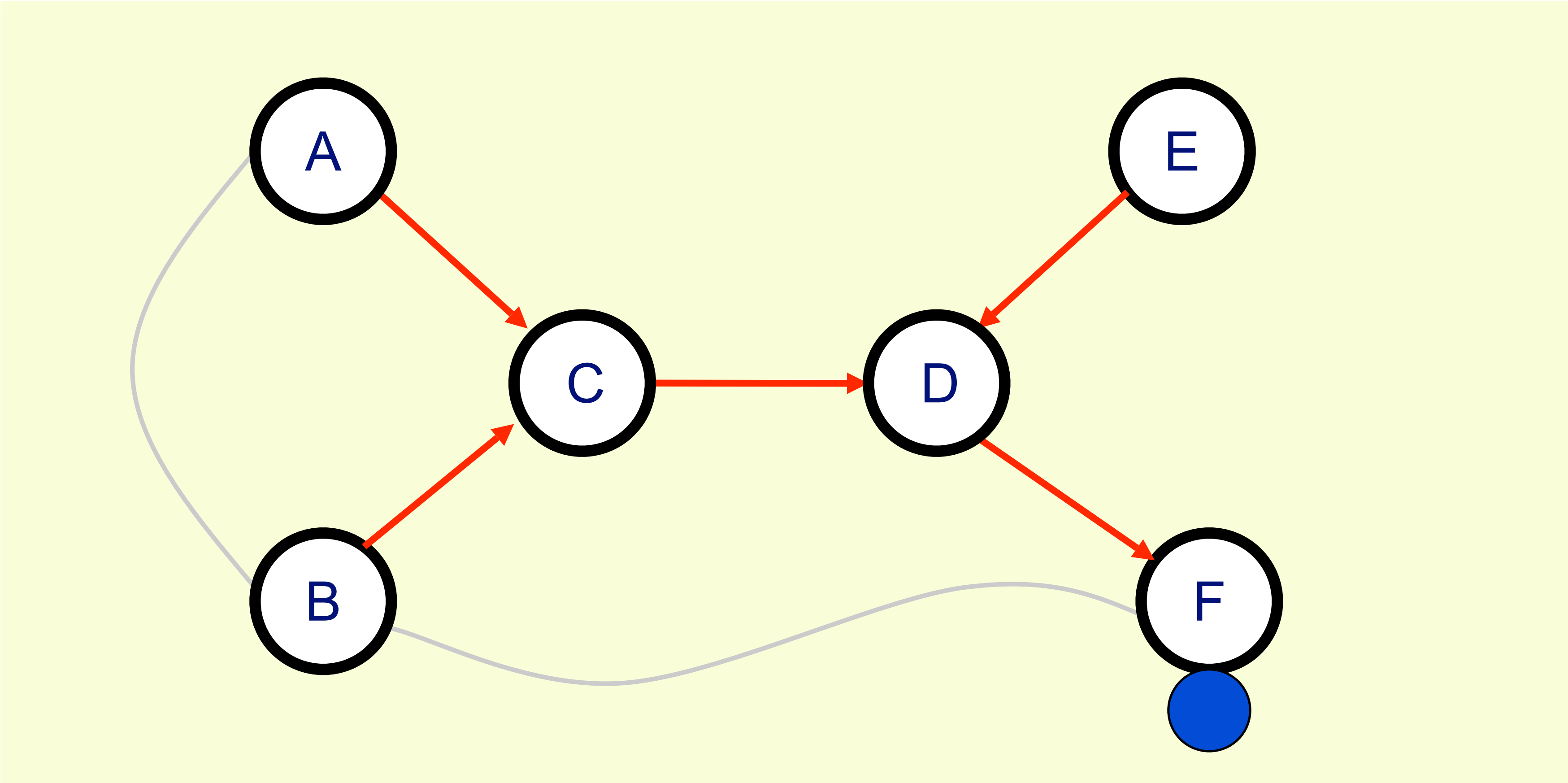
- "Some site crashed; the network is partitioned"

Spatial - Behavioral Properties



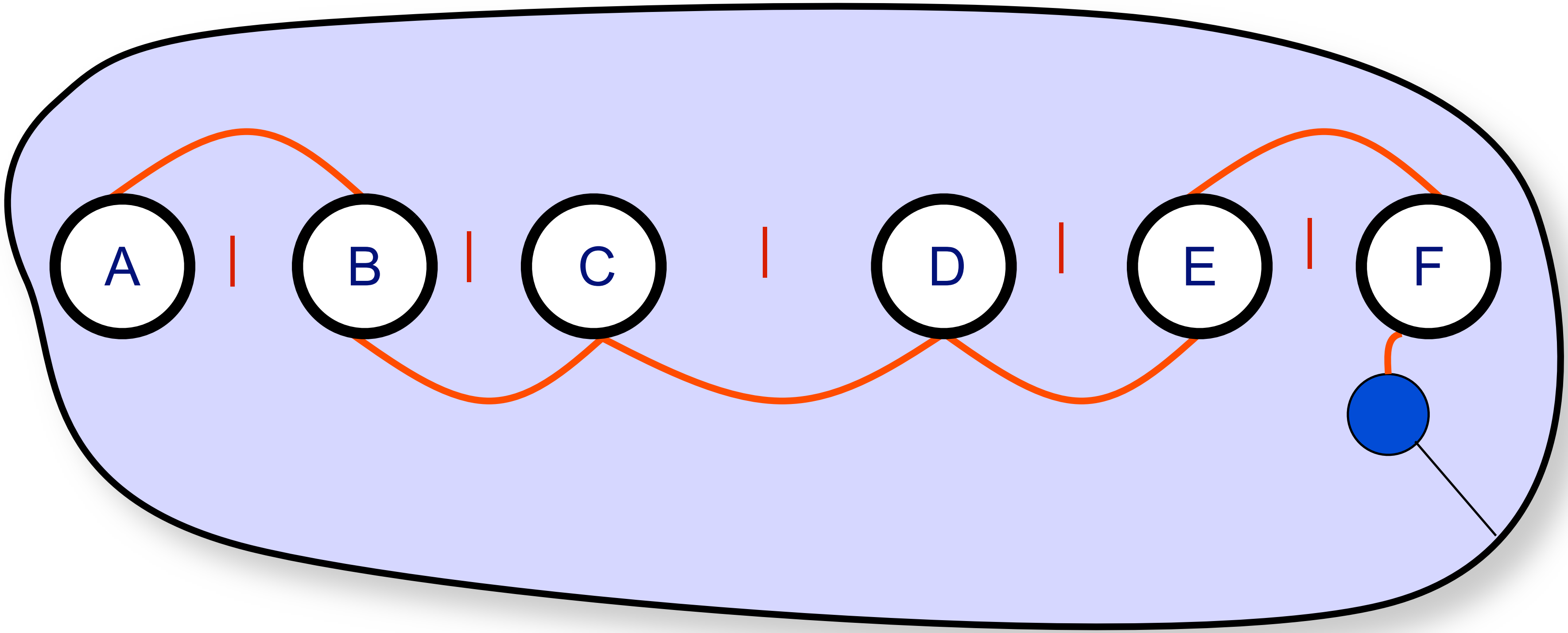
- “A new peer joined in”

Spatial - Behavioral Properties



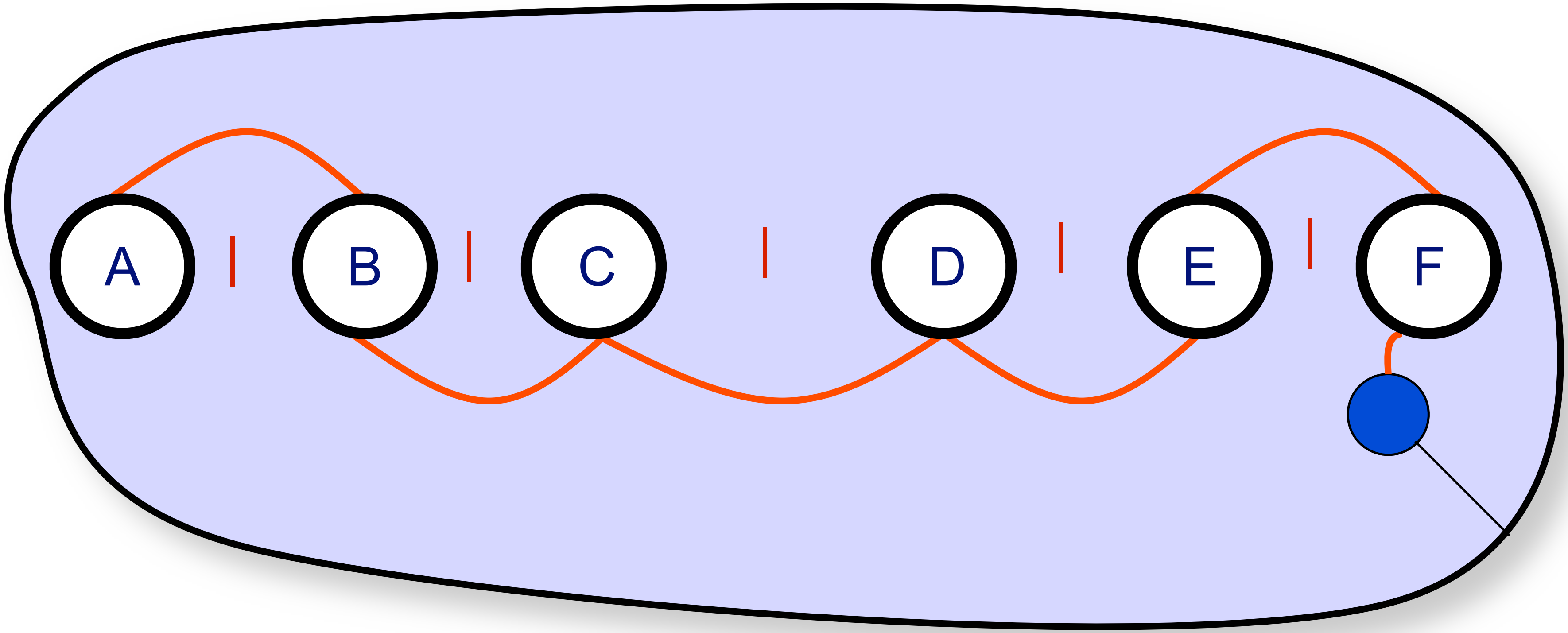
- “It is always possible for any site to eventually acquire exclusive access to the resource”

Basic Spatial Structure



- Concurrency (spatial) monoid <***Procs***, **0**, **|** >
- Spatial identity is (close to) structural congruence
- Name Restriction $(\nu n)P$

Basic Spatial Structure



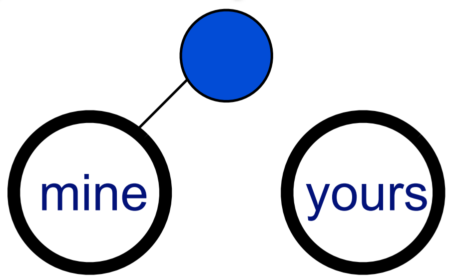
Directory $\triangleq$ (ν *obj*) (A | B | C | D | E | F)

Names and Spatial Structure

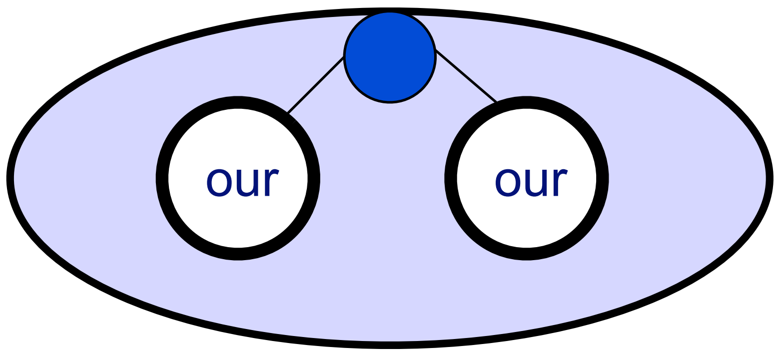
- Names identify resources in a (spatial) scope.
- Uses of names may be either public or hidden:

$$n \in fn(P) \quad \textit{if and only if} \quad \neg \exists Q.\ P \equiv (\nu n)Q$$

- A name always splits a system in two parts



- Hidden names can induce spatial bonds:

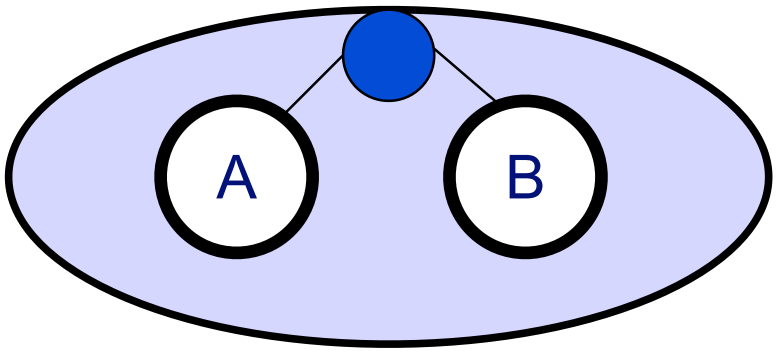


Names and Spatial Structure

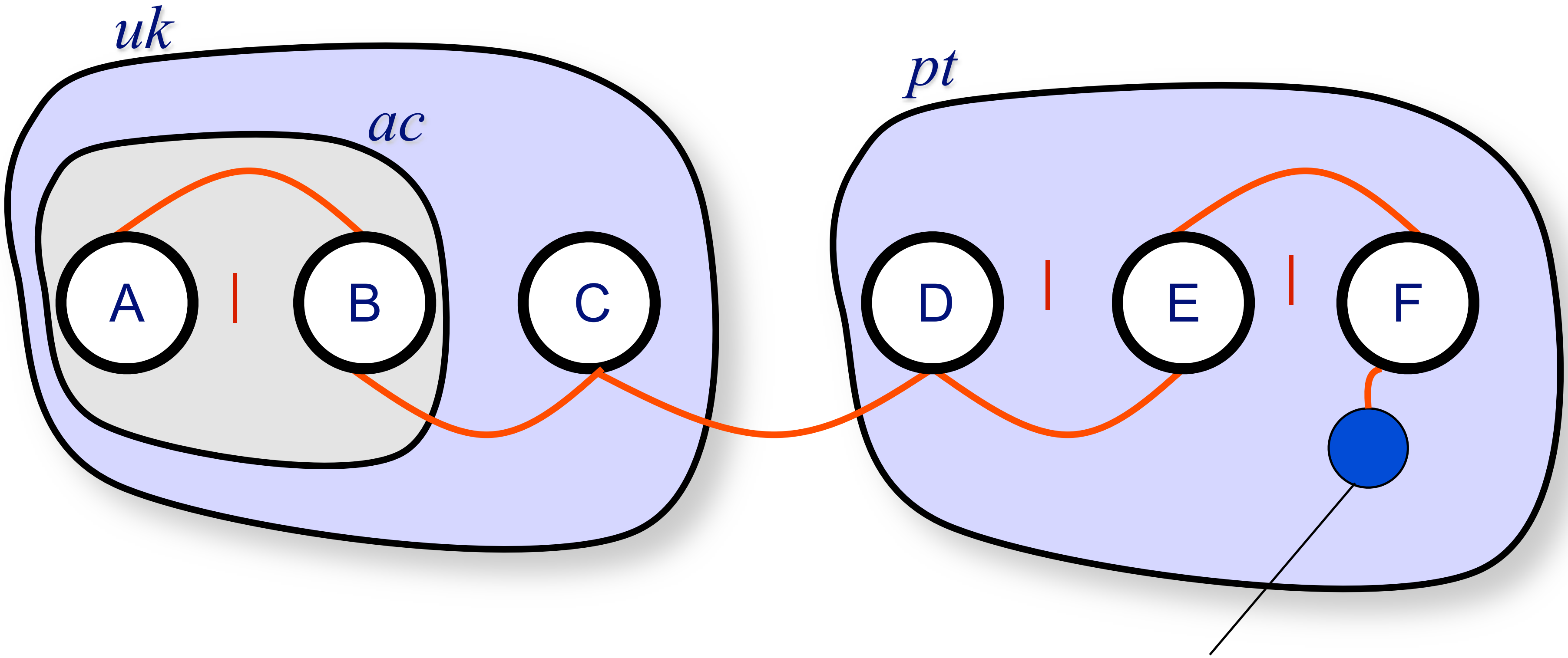
- *Pure names*, as construed by Roger Needham [N89], abstract general purpose atomic data.
- Names name resources, *e.g*., values, communication channels, secret keys, nonces, in a (spatial) scope.
- Uses of names may be either public or hidden:

$$n \in fn(P) \quad \textit{if and only if} \quad \neg\exists Q.\ P \equiv (\nu n)Q$$

- Hidden names may induce spatial bonds:



Nested Spatial Structure



- Concurrency (spatial) monoid <***Procs***, **0**, **|** >
- Tree structure (*cf.*, Mobile Ambients) : $n[\ P\ ]$
- Name Restriction: $(\nu n)P$
- Bigraphical Structure (Milner)

Properties of Distributed Models

- ## Temporal & Hennessy-Milner Logics
  - Modal logics with modalities for observing temporal structure
    - Useful for specifying general safety and liveness properties
    - Do not distinguish between bisimilar processes
- ## Spatial Logics [CM98,CG00,CC02-04,C04]
  - Modal logics with modalities for observing spatial structure
    - Each “world” is a structured space
    - “Space” is seen as a kind of resource (logics can separate, count)
- ## Spatial Observations
  - Not invariant under traditional behavioral equivalences
    - $n.\mathbf{0} + m.\mathbf{0} \neq_{\mathrm{L}} n.\mathbf{0} \mid m.\mathbf{0}$
    - Intensionality? (more later)
  - Invariant under a natural notion of spatial equivalence
    - $\approx$ structural congruence [San01]
    - $\approx$ extended structural congruence [Cai04]

The π-Calculus

n,m,p ∈ ***Names***

P,Q ∈ ***Procs*** ::= ***Processes***

$\mathbf{0}$	*Void*
$P \mid Q$	*Composition*
$(\nu n)P$	*Restriction*
$n!(m).P$	*Output*
$n?(m).P$	*Input*
$\Sigma\, \alpha_i.P_i$	*Choice*
$(\mathbf{rec}\ X[\mathbf{x}].P)[\boldsymbol{m}]$	*Recursion*
$X[\boldsymbol{m}]$	*Variable*

Reduction ($P \rightarrow Q$) :

$m!(n).P \mid m?(p).Q \rightarrow P \mid Q\{p/n\}$

$P \rightarrow Q$ *implies* $(\nu n)P \rightarrow (\nu n)Q$

$P \rightarrow Q$ *implies* $P \mid R \rightarrow Q \mid R$

$P \equiv P'$, $P' \rightarrow Q'$, $Q' \equiv Q$ *implies* $P \rightarrow Q$

Spatial Congruence:

$P \mid \mathbf{0} \equiv P$

$P \mid Q \equiv Q \mid P$

$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$

$(\nu n)\mathbf{0} \equiv \mathbf{0}$

$(\nu n)(\nu m)\ P \equiv (\nu m)(\nu n)P$

$(\nu n)(P \mid Q) \equiv P \mid (\nu n)Q$

if $n \notin fn(P)$

$(\mathbf{rec}\ X[\mathbf{x}].P)[\boldsymbol{m}] \equiv$

$P\ \{\mathbf{x} / \boldsymbol{m}\}\ \{X/\ (\mathbf{rec}\ X[\mathbf{x}].P)\}$

The π-Calculus

n,m,p ∈ ***Names***

$P, Q \in$ ***Procs*** ::= ***Processes***

$\mathbf{0}$	*Void*
$P \mid Q$	*Composition*
$(\nu n)P$	*Restriction*
$n!(m).P$	*Output*
$n?(m).P$	*Input*
$\Sigma\, \alpha_i.P_i$	*Choice*
$(\mathbf{rec}\ X[\mathbf{x}].P)[\boldsymbol{m}]$	*Recursion*
$X[\boldsymbol{m}]$	*Variable*

Interaction $(P \rightarrow Q)$:

$m, n \notin \boldsymbol{p}$

$$(\nu \boldsymbol{p})(n!(m).Q \mid P) \xrightarrow{n!(m)} (\nu \boldsymbol{p})(Q \mid P)$$

$m, n \notin \boldsymbol{p}$

$$(\nu \boldsymbol{p})(Q \mid n?(q).P) \xrightarrow{n?(m)} (\nu \boldsymbol{p})(Q|P\{q \leftarrow m\})$$

Spatial Congruence:

$$P \mid \mathbf{0} \equiv P$$

$$P \mid Q \equiv Q \mid P$$

$$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$$

$$(\nu n)\mathbf{0} \equiv \mathbf{0}$$

$$(\nu n)(\nu m)\, P \equiv (\nu m)(\nu n)P$$

$$(\nu n)(P \mid Q) \equiv P \mid (\nu n)Q \quad \textit{if } n \notin fn(P)$$

$$(\mathbf{rec}\ X[\mathbf{x}].P)[\boldsymbol{m}] \equiv P\,\{\mathbf{x} / \boldsymbol{m}\}\,\{X/\,(\mathbf{rec}\ X[\mathbf{x}].P)\}$$

Behavioral Observations

Processes behave by communicating:

$$n!(msg).Q \mid n?(x).P \xrightarrow{\tau} Q \mid P\{\,x/msg\,\}$$

Internal

$$P \xrightarrow{\tau} Q$$

Output

$$P \xrightarrow{n!(\,m\,)} Q$$

Input

$$P \xrightarrow{n?(\,m\,)} Q$$

$$\alpha ::= n!(m) \mid n?(m) \mid \tau$$

Hennessy-Milner like modalities:

$$\langle\alpha\rangle\, \mathcal{A}$$

$P \vDash \langle\alpha\rangle\, \mathcal{A}$ *iff* $P \xrightarrow{\alpha} Q$ *and* $Q \vDash \mathcal{A}$

Spatial Observations

- Composition and restriction are interpreted as spatial rather than dynamic operations. *E.g.*,
- Any process P can be decomposed in several ways into a pair (Q, R) such that the spatial identity holds:

$$P \equiv Q \mid R$$

$$P \overset{\|}{\rightarrow} (Q, R)$$

$\equiv$ "spatial congruence"

$$P \mid \mathbf{0} \equiv P$$

$$P \mid Q \equiv Q \mid P$$

$$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$$

Spatial Observations

- Composition and restriction are interpreted as spatial rather than dynamic operations. *E.g.*,
- Any process P can be decomposed in several ways into a pair ($\boldsymbol{n}$, Q) such that the spatial identity holds:

$$P \equiv (\nu \boldsymbol{n})Q$$

$$P \xrightarrow{\nu} (\boldsymbol{n}, Q)$$

$\equiv$ "spatial congruence"

$$(\nu n)\mathbf{0} \equiv \mathbf{0}$$

$$(\nu n)(\nu m)\, P \equiv (\nu m)(\nu n)P$$

$$(\nu n)(P \mid Q) \equiv P \mid (\nu n)Q$$

π-Calculus System Observations

- Composition

  $P \xrightarrow{\|} Q, R$

- Restriction

  $P \xrightarrow{\nu} n, Q$

- Internal Action

  $P \xrightarrow{\tau} Q$

- Input Action

  $P \xrightarrow{n?(m)} Q$

- Output Action

  $P \xrightarrow{n!(m)} Q$

$n,m \in$ ***Names***

$\alpha \in$ ***Actions*** ::=

τ	*Internal*
$n?(m)$	*Input Action*
$n!(m)$	*Output Action*

A core Spatial-Behavioral Logic

$P \models \mathcal{A}$

$\mathcal{A} \wedge \mathcal{B}$, $\neg\mathcal{A}$, ...	*Boolean Operators*
0	*Void*
$\mathcal{A} \mid \mathcal{B}$	*Composition*
$\mathsf{H}x.\mathcal{A}$	*Hidden Name Quantifier*
$@n$	*Free Name Occurrence*
$m = n$	*Name Equality*
$\langle\alpha\rangle\mathcal{A}$	*Action* ($\alpha \in$ ***Actions***)
$\forall x.\mathcal{A}$	*Universal Quantifier*
$\nu X.\mathcal{A}$	*Recursion (Greatest Fixed Point)*

Semantics

$P \vDash \mathbf{0}$ *iff* $P \equiv \mathbf{0}$

$P \vDash \mathcal{A} \mid \mathcal{B}$ *iff* $\exists Q, R.\ P \equiv Q \mid R$ and $Q \vDash \mathcal{A}$ and $R \vDash \mathcal{B}$

$P \vDash \mathsf{H}x.\mathcal{A}$ *iff* $\exists Q, n \# \mathcal{A}.\ P \equiv (\nu n)Q$ and $Q \vDash \mathcal{A}\{x/n\}$

$P \vDash @n$ *iff* $n \in fn(P)$

$P \vDash \langle \alpha \rangle \mathcal{A}$ *iff* $P \xrightarrow{\alpha} Q$ and $Q \vDash \mathcal{A}$

Semantics

$$
\begin{array}{lcl}
[\![\mathbf{True}]\!]_v & \triangleq & \textit{Procs} \\
[\![\mathcal{A} \wedge \mathcal{B}]\!]_v & \triangleq & [\![\mathcal{A}]\!]_v \cap [\![\mathcal{B}]\!]_v \\
[\![\neg \mathcal{A}]\!]_v & \triangleq & \textit{Procs} \setminus [\![\mathcal{A}]\!]_v \\
[\![m = n]\!]_v & \triangleq & \textbf{if } m = n \textbf{ then } \textit{Procs} \textbf{ else } \emptyset \\
[\![\mathbf{0}]\!]_v & \triangleq & \{ P \mid P \equiv \mathbf{0} \} \\
[\![\mathcal{A} \mid \mathcal{B}]\!]_v & \triangleq & \{ P \mid \exists Q,R.\ P \equiv Q \mid R \wedge Q \in [\![\mathcal{A}]\!]_v \wedge R \in [\![\mathcal{B}]\!]_v \} \\
[\![@n]\!]_v & \triangleq & \{ P \mid n \in fn(P) \} \\
[\![\mathsf{H}x.\mathcal{A}]\!]_v & \triangleq & \{ P \mid \exists Q.\ P \equiv (\nu n)Q \wedge n \notin fn_v(\mathcal{A}) \wedge Q \in [\![\mathcal{A}]\!]_v \} \\
[\![\langle\alpha\rangle\mathcal{A}]\!]_v & \triangleq & \{ P \mid \exists Q.\ P \xrightarrow{\alpha} Q \wedge Q \in [\![\mathcal{A}]\!]_v \} \\
[\![\forall x.\mathcal{A}]\!]_v & \triangleq & \bigcap n \in \textit{Names}.\ [\![\mathcal{A}\{x/n\}]\!]_v \\
[\![\mathcal{X}]\!]_v & \triangleq & v(\mathcal{X}) \\
[\![\nu\mathcal{X}.\mathcal{A}]\!]_v & \triangleq & \{ \psi \subseteq \textit{Procs} \mid \psi \subseteq [\![\mathcal{A}]\!]_{v[\mathcal{X} \leftarrow \psi]} \}
\end{array}
$$

Simple Examples

- $\mathcal{A}$ holds somewhere:

  $?\mathcal{A} \quad \triangleq \quad \mathcal{A} \mid \mathbf{True}$

- $\mathcal{A}$ holds everywhere:

  $!\mathcal{A} \quad \triangleq \quad \neg (\neg \mathcal{A} \mid \mathbf{True})$

- Has exactly one thread:

  $\mathbf{1} \quad \triangleq \quad \neg \mathbf{0} \wedge \neg (\neg \mathbf{0} \mid \neg \mathbf{0})$

  $? (\mathbf{1} \wedge \mathcal{A})$ — $\mathcal{A}$ holds of some thread

  $\mathcal{A}^* \quad \triangleq \quad ! (\mathbf{1} \Rightarrow \mathcal{A})$ — $\mathcal{A}$ holds of every thread

- Arithmetic constraints on the number of threads

  $\mathbf{gt}(n) \quad \triangleq$

- After an arbitrary step:

  $\boxdot\mathcal{A} \quad \triangleq \quad [\, \tau \,]\mathcal{A} \vee [\, ? \,]\mathcal{A} \vee [\, ! \,]\mathcal{A}$

- Always in the future:

  $\boxdot^* \mathcal{A} \quad \triangleq \quad \nu X. (\mathcal{A} \wedge \boxdot X)$

Simple Examples

- Uses an hidden name x, that satisfies P(x)

  Hx. (P(x) $\wedge$ ©x)   N.B.: $\text{И}x.\mathcal{A} \triangleq$ Hx. ($\mathcal{A} \wedge \neg$ © x)

- Creating bonds through hidden names:

  Let $P \triangleq$ ($(\nu n)\ m!(n).n?(p).Q$) | $m?(q).q!(q).R$

  Then $P \vDash$ ($\neg$ **0** | $\neg$ **0**) $\wedge \langle\tau\rangle$ **1**

  and $P \vDash$ $\Diamond$ Hx.(© x | © x)

  $\Diamond\mathcal{A} \triangleq \langle\tau\rangle\mathcal{A}$

- Keeps no secrets:

  Public $\triangleq$ $\neg$ Hx.© x

- A holds inside (insider knows all secrets, but does not tell):

  inside(A) $\triangleq$ μX. ((Public $\wedge$ $\mathcal{A}$) $\vee$ Hx. (© x $\wedge$ X))

  N.B.: $P \vDash$ $\neg$ inside($\mathcal{A}$) *iff* $P \vDash$ inside($\neg\mathcal{A}$)

The Freshness Quantifier (cf. Gabbay-Pitts)

The freshness quantifier $\text{И}x.\mathcal{A}$ is defined such that a process P satisfies $\text{И}x.\mathcal{A}$ if and only if P satisfies $\mathcal{A}\{x/n\}$ for some name n fresh in P and in $\mathcal{A}$.

$$P \vDash_v \text{И}x.\mathcal{A} \quad \textit{iff} \quad \exists n \in \mathcal{N}.\ n \notin fn^v(P, \mathcal{A}) \textit{ and } P \vDash_v \mathcal{A}\{x/n\}$$

$$P \vDash_v \text{И}x.\mathrm{A} \quad \textit{iff} \quad \forall n \in \mathcal{N}.\ n \notin fn^v(P,\mathcal{A}) \textit{ implies } P \vDash_v \mathcal{A}\{x/n\}$$

(A property true of some fresh name is true of any fresh name)

Some properties of the fresh name quantifier:

$$\forall x.\mathcal{A} \vdash \text{И}x.\mathcal{A} \vdash \exists x.\mathcal{A}$$

$$\neg\, \text{И}x.\mathcal{A} \dashv\vdash \text{И}x.\, \neg\mathcal{A}$$

$$\text{И}x.\, \mathcal{A} \mid \text{И}x.\mathcal{B} \dashv\vdash \text{И}x.(\mathcal{A} \mid \mathcal{B})$$

$$\text{И}x.\Diamond\mathcal{A} \dashv\vdash \Diamond\text{И}x.\mathcal{A}$$

Some Properties of $\mathsf{H}x.A$

- "Irrelevant" hidden name quantification reduces to freshness quantification:

$$\mathsf{H}x.(\mathcal{A} \land \neg ©x) \dashv\vdash \text{И}x.\, \mathcal{A}$$

- Logical characterisations of scope extrusion:

$$(\mathsf{H}x.\mathcal{A}) \mid \mathrm{B} \dashv\vdash \mathsf{H}x.\, (\mathcal{A} \mid \mathcal{B} \land \neg ©x)$$

$$(\mathsf{H}x.\mathcal{A}) \mid (\forall x.\mathcal{B}) \vdash \mathsf{H}x.(\mathcal{A} \mid \mathcal{B})$$

- Some "inversion" principles:

$$\mathsf{H}x.\Diamond\mathcal{A} \dashv\vdash \Diamond\mathsf{H}x.\mathcal{A}$$

$$(\mathsf{H}x.\mathcal{A}) \land (\mathsf{H}x.\mathcal{B}) \dashv\vdash \mathsf{H}x.(\mathcal{A}\land\mathcal{B}) \lor \mathsf{H}x.\mathsf{H}y.(\mathcal{A}\land\mathcal{B}\{x/y\})$$

Resource Control and Secrecy

- Spatial implication

  $\mathcal{A} \blacktriangleright \mathcal{B} \triangleq (\neg\, \mathcal{A}) \,||\, \mathcal{B}$

- Unique handling of requests

  $\boxdot^{*}\,(\text{inside} \neg\, \exists y.\ (\exists x.\ \langle\, y?(x)\,\rangle \mathbf{True} \mid \exists x.\ \langle\, y?(x)\,\rangle \mathbf{True}\ ))\ )$

- Resource control (race freedom):

  $\boxdot^{*}\,(\text{inside} \neg\, \exists y.\ (\ \exists x.\ \langle\, y!(x)\,\rangle \mathbf{True} \mid$

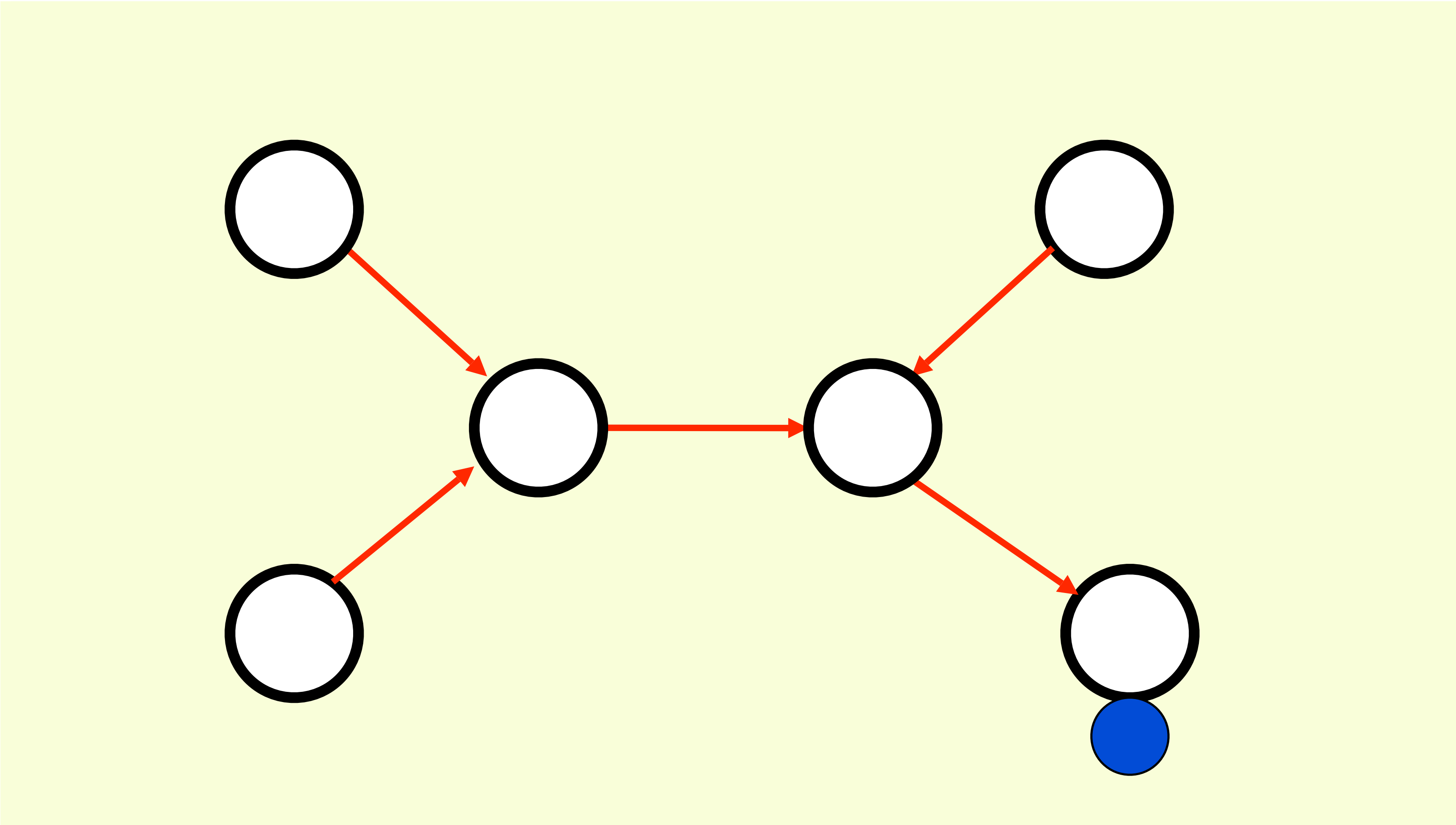
  $\exists x.\ \langle\, y!(x)\,\rangle \mathbf{True} \mid \exists x.\ \langle\, y?(x)\,\rangle \mathbf{True}\ ))\ )$

- Secrecy:

  $\boxdot^{*}\,(\ \neg\, \exists y.\ \text{H} x.(\ \mathcal{A}(x) \wedge \langle\, y!(x)\,\rangle \mathbf{True}\ )\ )$

  $\mathcal{A}(x) \triangleq \exists y.\ x(y).\mathbf{True}$ (never leaks private resources)

Spatial - Behavioral Properties



- "It is always possible for any site to eventually acquire exclusive access to the resource"

Spatial - Behavioral Properties

- "It is always possible for any site to eventually acquire exclusive access to the resource"

$$\mathrm{beh}(n) \triangleq \ldots$$

$$\mathrm{node}(n) \triangleq \mathbf{1} \wedge \mathrm{beh}(n)$$

$$\mathrm{owns}(n, x) \triangleq \mathrm{node}(n) \wedge \copyright\, x$$

$$\mathrm{exclusive}(n, x) \triangleq (\ \mathrm{owns}(n, x) \mid \neg \copyright\, x\ )$$

$$\mathrm{live} \triangleq \mathrm{H}x.\ \mathrm{inside}(\ \mathrm{obj}(x) \mid \forall n.\ ?\mathrm{node}(n) \Rightarrow \mathrm{eventually}\ (\mathrm{exclusive}(n, x)))$$

$$\mathrm{Safety} \triangleq \mathrm{always}\ (\ \mathrm{live}\ )$$

Adjunct Operators

- Minimal contextual observations (*cf.*, labeled transitions) are not that easy to define in general.
- The composition adjunct operator, introduced in the Ambient Logic [CG00], allows context dependent properties to be defined in a general way (*cf.*, barbed equivalence).
- The composition adjunct (guarantee)

$\mathcal{A} \mid \mathcal{B}$ *Composition*

$\mathcal{A} \triangleright \mathcal{B}$ *Guarantee*

$$P \vDash \mathcal{A} \triangleright \mathcal{B} \textit{ iff } \forall Q.\ \textit{if } Q \vDash \mathcal{A} \textit{ then } P \mid Q \vDash \mathcal{B}$$

- The logical equivalence induced by a spatial logic containing just the adjunct operators (and not the "basic" spatial operators) is strong bisimilarity [H04] (on finite processes).

Specification of a Simple Protocol

$\text{Client} \triangleq \ \mathsf{H}x.\ (\ \text{Auth}(x) \mid \text{Request}(x)\ )$

$\text{Server} \triangleq \ \nu Y.\ \text{И}x.\ \text{Auth}(x) \triangleright \ \Diamond\ (\ \text{Handler}(x) \mid Y\ )$

$\text{Auth}(x) \triangleq$ … specification of the authentication protocol …

By unfolding we get:

$\text{Server} \dashv\vdash \text{И}x.\ \text{Auth}(x) \triangleright \Diamond\ (\ \text{Handler}(x) \mid \text{Server}\ )$

We can then prove:

$\text{Server} \mid \text{Client} \vdash \Diamond\ (\ \text{Server} \mid \mathsf{H}x.(\text{Handler}(x) \mid \text{Request}(x)))$

Expressiveness of Adjunct

- Adjunct allows an internal definition of validity [CG00]

  valid($\mathcal{A}$) $\triangleq$ ($\neg\mathcal{A}$) $\triangleright$ False   satisfiable($\mathcal{A}$) $\triangleq$ $\mathcal{A} \triangleright$ True

  We have: $P \models$ valid($\mathcal{A}$) *iff* $\forall Q.\ Q \models \mathcal{A}$

- Validity and model-checking of static quantifier-free spatial logics with adjunct is decidable [CYOH01,CCG03].
- Composition adjunct does not add to the expressiveness of static quantifier free spatial logics [L03] (it does in most other fragments).
- Validity and model-checking of spatial logics with adjunct and existential name quantifier is undecidable [CT02].
- Validity and model-checking of spatial logics with adjunct and hidden name quantifier is undecidable [CG04].
- Validity and model-checking of spatial logics with adjunct and nextstep is undecidable [CL04] (can encode first-order logic).

Behavioral vs. Spatial Observations

- Basic behavioral and spatial observations look quite elementary. However, the combination of behavioral and spatial properties turns out to be very expressive.
- Behavioral observations are definable in pure spatial logics, exploiting the adjunct [S01,CC01,HLS03,H04].
- Logics with spatial observations and adjunct:
  - More convenient for compositional reasoning.
  - Model and validity-checking is undecidable and incomplete.
- Logics with behavioral and spatial observations:
  - Still expressive...
  - Model-checking is decidable and complete [C04].

Model-Checking

Model-Checking

Main technical issues:

Handling name creation and freshness
Recursion in the presence of freshness
Operations on Processes w.r.t. structural congruence

Decidability of Structural Congruence.

For all processes P and Q, we can decide whether $Q \equiv P$.

For any finite set of names M, we can decide whether $Q \equiv_M P$.

Key to the Completeness Proof.

A coinductive characterization of extended structural congruence (cf. spatial bisimulation).

A “unique solution” theorem for systems of process equations modulo structural congruence (cf. Amadio-Cardelli).

Model-Checker

$\texttt{MCheck}$: *Procs* $\times$ *SVal* $\times\ \Phi \rightarrow$ bool

$\texttt{MCheck}\,(P,\ v,\ \mathcal{A})$ assumes *Dom*$(v) \subseteq$ *fpv*$(\mathcal{A})$

$\texttt{MCheck}\,(P,\ v,\ \mathbf{T}) \triangleq \texttt{true}$

$\texttt{MCheck}\,(P,\ v,\ \neg\mathcal{A}) \triangleq \texttt{not}\ \texttt{MCheck}\,(P,\ v,\ \mathcal{A})$

$\texttt{MCheck}\,(P,\ v,\ \mathcal{A}\wedge\mathcal{B}) \triangleq \texttt{MCheck}\,(P,\ v,\ \mathcal{A})\ \texttt{\&}\ \texttt{MCheck}\,(P,v,\mathcal{B})$

$\texttt{MCheck}\,(P,\ v,\ \mathbf{0}) \triangleq \texttt{Check}\,(P \equiv \mathbf{0})$

$\texttt{MCheck}\,(P,\ v,\ \mathcal{A}\,|\,\mathcal{B}) \triangleq \texttt{Exists}\ \langle Q,R\rangle \in \mathit{Comp}(P).$

$\texttt{MCheck}\,(Q,\ v,\ \mathcal{A})\ \texttt{\&}\ \texttt{MCheck}\,(R,\ v,\mathcal{B})$

$\texttt{MCheck}\,(P,\ v,\ @n) \triangleq \texttt{Check}(\mathit{Res}(P,\ n) \neq \emptyset)$

$\texttt{MCheck}\,(P,\ v,\ \langle\alpha\rangle\mathcal{A}) \triangleq \texttt{Exists}\ Q \in \mathit{Comm}(P,\alpha).\texttt{MCheck}\,(Q,v,\mathcal{A})$

Model-Checking: Process Observations

Composition. For every process P we can compute a finite set of pairs of processes *Comp*(P) such that:

$\langle Q,R\rangle \in Comp(P)$ implies $P \equiv Q \mid R$

$P \equiv Q \mid R$ implies $\exists \langle Q,R\rangle \in Comp(P).\ Q \equiv Q' \wedge R \equiv R'$

N.B.: *Comp*(P) is finite because we don't have $!P \equiv !P \mid P$.

Name Restriction. For every process P and name n we can compute a finite set of processes *Res*(P,n) such that:

$Q \in Res(P,n)$ implies $P \equiv (\nu n)Q$

$P \equiv (\nu n)Q$ implies $\exists R \in Res(P,n).\ R \equiv Q$

N.B.: *Res*(P,n) $= \emptyset$ if and only if $n \in fn(P)$

Commitment. For every process P and action α we can compute a finite set of processes *Comm*(P,α) such that:

$Q \in Comm(P,\alpha)$ implies $P \xrightarrow{\alpha} Q$

$P \xrightarrow{\alpha} Q$ implies $\exists R \in Comm(P,\alpha).\ R \equiv Q$

Model-Checking: Syntactic Valuations

Syntactic Valuation.

A ***syntactic valuation*** ($v \in$ ***SVal***) is a sequence

$$[X_1 \rightarrow (\mathcal{S}_1, \mathcal{A}_1)] \ldots [X_n \rightarrow (\mathcal{S}_n, \mathcal{A}_n)]$$

of assignments such that:

Each X_i is a propositional variable.

Each $\mathcal{A}_i$ is a fixpoint formula of the form $\nu X_i.\mathcal{B}_i$.

$fpv(A_i) \subseteq \{ X_1, \ldots, X_{i-1} \}$

Free Names of formula $\mathcal{A}$ under syntactic valuation v.

$$fs^v(\mathcal{A}) \triangleq \bigcup \{ fs^v(\mathcal{B}) \mid v(X) = (\mathcal{S}, \mathcal{B}) \wedge X \in fpv(\mathcal{A}) \} \cup fn(\mathcal{A})$$

c.f., $fn^v(\mathcal{A}) \triangleq \bigcup \{ supp(v(X)) \mid X \in fpv(\mathcal{A}) \} \cup fn(A)$

Model-Checker

MCheck: *Procs* × *SVal* × $\Phi \to$ bool

MCheck $(P, v, \mathcal{A})$ assumes $Dom(v) \subseteq fpv(\mathcal{A})$

MCheck $(P, v, \forall x.\mathcal{A}) \triangleq$

let $M = fs^{v}(\forall x.\mathcal{A}) \cup fn(P)$

in **Forall** $n \in M \cup \{ \; fresh(M) \}$. **MCheck** $(P, v, \mathcal{A}\{x/n\})$

MCheck $(P, v, \text{H}x.\mathcal{A}) \triangleq$

let $n = fresh(fs^{v}(\text{H}x.\mathcal{A}))$ **and** $Q \in Res(P,n)$

in **MCheck** $(Q, v, \mathcal{A}\{x/n\})$

N.B. *fresh*(*M*) picks some name **out** of the finite set *M*.
N.B. Any *fresh*(-) function can be used (*e.g.*, ad-hoc gensym).

Model-Checker

$\texttt{MCheck}: \mathit{Procs} \times \mathit{SVal} \times \Phi \rightarrow \text{bool}$

$\texttt{MCheck}(P, v, \mathcal{A})$ assumes $\mathit{Dom}(v) \subseteq \mathit{fpv}(\mathcal{A})$

$\texttt{MCheck}(P, v, \nu X.\mathcal{A}) \triangleq \texttt{MCheck}(P, v[X \mapsto (\{P\}, \nu X.\mathcal{A})], \mathcal{A})$

$\texttt{MCheck}(P, v, X) \triangleq$

$\quad \texttt{if In}(P, v, X)\ \texttt{then true}$

$\quad\quad \texttt{else let}\ (\mathcal{S}, \nu X.\mathcal{G}) = v(X)$

$\quad\quad\quad \texttt{in MCheck}(P, v[X \mapsto (\mathcal{S} \cup \{P\}, \nu X.\mathcal{G})], \mathcal{G})$

$\texttt{In}(P, v, X) \triangleq \texttt{let}\ (\mathcal{S}, \mathcal{F}) = v(X)$

$\quad\quad \texttt{and}\ M = \mathit{fs}^{v}(\mathcal{F})$

$\quad\quad\quad \texttt{in Exists}\ Q \in \mathcal{S}.\, Q \equiv_{M} P$

Soundness of the Model-Checker

Semantic valuation. For any (syntactic) valuation $v = w[X \to (\mathcal{S}, \nu X.\mathcal{A})]$ we define a corresponding (semantic) valuation v^* by letting

$$v^*(X) \triangleq \textit{Gfix}(\lambda \mathcal{V}.\ \textit{Clos}(\mathcal{S}, M) \cup [\![\mathcal{A}]\!]_{w^*[X \leftarrow \mathcal{V}]})$$

wherever $v(X) = (\mathcal{S}, \nu X.\mathcal{A})$ and $M = fs^v(\nu X.\mathcal{A})$

Soundness. If `MCheck(`P, v, $\mathcal{A}$`) = true` then $P \in [\![\mathcal{A}]\!]_{v^*}$

If `MCheck(`P, v, $\mathcal{A}$`) = false` then $P \notin [\![\mathcal{A}]\!]_{v^*}$

Key to the Proof.

Support & Closure.

$supp([\![\mathcal{A}]\!]_{v^*}) \subseteq fn^{v^*}(\mathcal{A}) \subseteq fs^v(\mathcal{A})$

If $P \in [\![\mathcal{A}]\!]_v$ and $fn^v(\mathcal{A}) \subseteq M$ and $P \equiv_M Q$ then $Q \in [\![\mathcal{A}]\!]_v$

Kozen-Winskel.

Define the mapping ϕ such that $\phi(\mathcal{S}) \triangleq [\![\mathcal{A}]\!]_{v[X \leftarrow \mathcal{S}]}$. Then:

For all $\psi \subseteq$ ***Procs***, $\psi \subseteq \textit{Gfix}(\phi)$ iff $\psi \subseteq \phi(\textit{Gfix}(\lambda \mathcal{S}.\phi(\psi \cup \mathcal{S})))$

Completeness of the Model-Checker

Reachability.

$P \in \mathit{Reach}(P)$

$P \in \mathit{Reach}(P),\ Q \in \mathit{Com}(P,\alpha) \Rightarrow Q \in \mathit{Reach}(P)$

$P \in \mathit{Reach}(P),\ \langle Q,R \rangle \in \mathit{Comp}(P) \Rightarrow Q, R \in \mathit{Reach}(P)$

$P \in \mathit{Reach}(P),\ Q \in \mathit{Res}(P,n) \Rightarrow Q \in \mathit{Reach}(P)$

Bounded. A process P is *bounded* if for every finite set of names M, the set of equivalence classes $\mathit{Reach}(P) / \equiv_M$ is finite.

N.B.: All recursion-free processes are bounded.

All finite-control processes are bounded.

It is not the case that every terminating process is bounded.

Completeness & Decidability.

If P is bounded and $P \in [\![\mathcal{A}]\!]_{v^*}$ then `MCheck(`P, v, $\mathcal{A}$`) = true`

The Spatial Logic Model Checker [VC04,05,06]

- Developed in UNLisbon, based on the techniques described above [Cai04] (complete for “bounded” processes).
- On-the-fly state-space generation.
- Ocaml implementation (available in source form).
- Supports the full π-calculus with parametric recursion.
- Supports a full adjunct-free logic with parametric recursion.
- Version 1.1 is available on the web:

  `http://ctp.di.fct.unl.pt/SLMC/`

- You may find some worked out examples there.

Expressiveness and Intensionality

Degrees of Observational Power

- The logical equivalence induced by spatial logics is quite sensitive to the presence of particular process operators, logical operators, and to the presentation of structural congruence. *E.g.*,

  Sangiorgi has shown, in seminal work [San01], that for the ambient logic and finite public ambient calculus, $=_L$ coincides with $\equiv$ (intensionality).

  If we consider the choice-free finite π-calculus and the core spatial logic, we also obtain $\equiv\; = \;=_L$.

  However, if we add choice or recursion, then we have $\equiv\; \subset \;=_L$.

  In a certain distributed calculus (below), if the semantics is crafted so that a single site may fail (at each reduction step) instead of an arbitrary subsystem, then strong bisimilarity collapses to spatial bisimilarity.

- It is then important to study "principled" spatial models and spatial logics, for which logical equivalence coincides with isomorphism of (the intended / observable) spatial structure.

Characterisation of $=_L$ in the π-calculus

- Logical Equivalence of Processes ($=_L$)

$$P =_L Q \quad \triangleq \quad \forall \mathcal{A}.\ (\ P \models \mathcal{A} \ \textit{iff}\ Q \models \mathcal{A}\ )$$

- Axiomatization of logical equivalence on the full pi-calculus with choice and recursion [Cai04]:

  $=_L$ is the least congruence relation $\equiv^{\varepsilon}$ on processes such that:

  $P \equiv Q$ implies $P \equiv^{\varepsilon} Q$

  $P + P \equiv^{\varepsilon} P$

  cf. idempotence of ~

  $P \equiv^{\varepsilon} Q\{Y / (x)P\}$ implies $P \equiv^{\varepsilon}$ (**rec** $Y[x].Q)[$

  co-induction

  We have: Both $\equiv^{\varepsilon}$ and $=_L$ are decidable.

  We have: $\equiv \ \subset \ =_L \ \subset \ \sim$ (strict inclusions)

Spatial Bisimulation (π-calculus)

- Caracterizes indistinguishable states (coinductively)

A binary relation $\mathcal{B} \subseteq \mathcal{P} \times \mathcal{P}$ is a spatial bisimulation if

for all $(s,r) \in \mathcal{B}$

if $s \xrightarrow{\alpha} s'$ then there is r' s.t. $r \xrightarrow{\alpha} r'$ and $(s',r') \in \mathcal{B}$

(and conversely)

if $s \equiv \mathbf{0}$ then $r \equiv \mathbf{0}$

(and conversely)

if $s \equiv p \mid q$ for some p,q then there are u,v s.t. $r \equiv u \mid v$

and $(p,u) \in \mathcal{B}$ and $(p,u) \in \mathcal{B}$

(and conversely)

if $s \equiv (\nu n)p$ for some p,n exists u s.t. $r \equiv (\nu n)u$ and $(p,u) \in \mathcal{B}$

(and conversely)

- Spatial bisimilarity $\sim_s$: the greatest spatial bisimulation

$$\sim_{s0} = \bigcup \{ \mathcal{B} \mid \mathcal{B} \text{ is a spatial bisimulation } \}$$

Spatial Bisimulation (π-calculus)

- We have

  $n!() \mid m!() \not\sim^s n!().m!()+m!().n!()$

  although

  $n!() \mid m!() \sim n!().m!()+m!().n!()$

  (cf. "true concurrency" semantics)
- We have

  $0 \not\sim^s (\nu n)n!()$

  although

  $0 \sim (\nu n)n!()$

  ($\sim^s$ distinguishes deadlock from proper termination)
- If P,Q are sequential (no parallel composition) and public (no restricted names) then $P \sim Q$ implies $P \sim^s Q$

Extensionality in Spatial Observations

- In general, we may expect spatial observations, as captured by a spatial logic SL, to induce a degree of intensionality, in the sense that logical equivalence is strictly finer than behavioral equivalence.
- However, purely behavioral equivalences in process calculi with spatial constructs (*e.g.*, mobile ambients) are already fairly sensitive to system properties usually considered “intensional”, such as arithmetical constraints in the number of sites, etc ...
- For example, in Sangiorgi [S01] has shown that for the public ambient calculus and ambient logic,

$$\equiv \ (=) \ =_{SL} \ (\subset) \ \approx$$

But how far is $=_{SL}$ from $\sim$ (and $\approx$)?

There are natural models where $=_{SL} \ (=) \ \approx$

Extensionality in Spatial Observations

A minimal distributed model

A set $\mathcal{A}$ of actions (α, τ) where $\alpha = n$ or $\alpha = \overline{n}$

A set $\mathcal{P}$ of processes

$$P, Q, R ::= \mathbf{nil} \mid P \mid Q \mid \alpha.P \mid \mathbf{go}.P$$

A set $\mathcal{N}$ of networks

$$N, M, S ::= \mathbf{0} \mid [\,P\,] \mid M \mid N$$

Structural congruence $\equiv$

$$P \mid \mathbf{nil} \equiv P \qquad N \mid \mathbf{0} \equiv N$$

$$P \mid Q \equiv Q \mid P \qquad N \mid M \equiv M \mid N$$

$$(P \mid Q) \mid R \equiv P \mid (Q \mid R) \qquad (N \mid M) \mid S \equiv N \mid (M \mid S)$$

$$\frac{P \equiv Q}{[\,P\,] \equiv [\,Q\,]}$$

Extensionality in Spatial Observations

A minimal distributed model

A set $\mathcal{A}$ of actions (α, τ) where $\alpha = \mathrm{n}$ or $\alpha = \overline{\mathrm{n}}$

A set $\mathcal{P}$ of processes

$$P, Q, R ::= \mathbf{nil} \mid P \mid Q \mid \alpha.P \mid \mathbf{go}.P$$

A set $\mathcal{N}$ of networks

$$N, M, S ::= \mathbf{0} \mid [\, P \,] \mid M \mid N$$

Reduction $\rightarrow$

$$[\, \alpha.P \mid \overline{\alpha}.Q \mid R \,] \rightarrow [\, P \mid Q \mid R \,]$$

$$[\, \tau.P \mid R \,] \rightarrow [\, P \mid R \,]$$

$$[\, \mathbf{go}.P \mid R \,] \mid [\, Q \,] \rightarrow [\, R \,] \mid [\, P \mid Q \,]$$

$$[\, P \,] \mid N \rightarrow \mathbf{0}$$

$$\frac{M \equiv M' \quad M' \rightarrow N' \quad N' \equiv N}{M \rightarrow N}$$

$$\frac{M \rightarrow N}{M \mid S \rightarrow N \mid S}$$

Extensionality in Spatial Observations

A minimal distributed model

A set $\mathcal{A}$ of actions (α, τ) where $\alpha = \mathrm{n}$ or $\alpha = \overline{\mathrm{n}}$

A set $\mathcal{P}$ of processes

$$P, Q, R ::= \mathbf{nil} \mid P \mid Q \mid \alpha.P \mid \mathbf{go}.P$$

A set $\mathcal{N}$ of networks

$$N, M, S ::= \mathbf{0} \mid [\, P \,] \mid M \mid N$$

Reduction $\rightarrow$

$$[\, \alpha.P \mid \overline{\alpha}.Q \mid R \,] \rightarrow [\, P \mid Q \mid R \,]$$

$$[\, \tau.P \mid R \,] \rightarrow [\, P \mid R \,]$$

migration

$$[\, \mathbf{go}.P \mid R \,] \mid [\, Q \,] \rightarrow [\, R \,] \mid [\, P \mid Q \,]$$

$$[\, P \,] \mid N \rightarrow \mathbf{0}$$

failure

$$\frac{M \equiv M' \quad M' \rightarrow N' \quad N' \equiv N}{M \rightarrow N}$$

$$\frac{M \rightarrow N}{M \mid S \rightarrow N \mid S}$$

Extensionality in Spatial Observations

- A minimal distributed model

  A set $\mathcal{A}$ of actions (α, τ) where $\alpha = n$ or $\alpha = \overline{n}$

  A set $\mathcal{P}$ of processes

$$P, Q, R ::= \mathbf{nil} \mid P \mid Q \mid \alpha.P \mid \mathbf{go}.P$$

  A set $\mathcal{N}$ of networks

$$N, M, S ::= \mathbf{0} \mid [\, P \,] \mid M \mid N$$

- Barb observation

$$[\, \alpha.P \mid Q \,] \downarrow \alpha$$

- We then pick observational equivalence for networks (undistinguishability) to be strong reduction barbed congruence $\simeq$ defined in the standard way.

Extensionality in Spatial Observations

- A minimal distributed model

  A set $\mathcal{A}$ of actions (α, τ) where $\alpha = n$ or $\alpha = \overline{n}$

  A set $\mathcal{P}$ of processes

$$P, Q, R ::= \mathbf{nil} \mid P \mid Q \mid \alpha.P \mid \mathbf{go}.P$$

  A set $\mathcal{N}$ of networks

$$N, M, S ::= \mathbf{0} \mid [\, P \,] \mid M \mid N$$

- Labels

$$\lambda ::= \tau \mid \alpha \mid \overline{\alpha} \mid [\alpha]$$

- Labeled observations $\xrightarrow{\lambda}$

$$[\, \overline{\alpha}.P \mid Q \,] \xrightarrow{\overline{\alpha}} [\, P \mid Q \,]$$

$$[\, \alpha.P \mid Q \,] \xrightarrow{\alpha} [\, P \mid Q \,] \qquad\qquad N \xrightarrow{[\alpha]} N \mid [\, \alpha.\mathbf{nil} \,]$$

Extensionality in Spatial Observations

- Minimal Spatial Logic

$$\mathcal{A}, \mathcal{B}, \mathcal{C} ::= \text{True} \mid \mathcal{A} \wedge \mathcal{B} \mid \neg\mathcal{A} \mid \mathbf{0} \mid \mathcal{A} | \mathcal{B} \mid \langle\lambda\rangle\mathcal{A}$$

- Theorem (Characterization) [CaiVie06]

$$M \sim N \text{ if and only if } M =_L N$$

- Theorem (Minimality)

In the model considered, no proper fragment of the minimal spatial logic preserves its separation power.

- Moral

Spatial observations are not necessarily intensional or arbitrary. In distributed systems, computational contexts can already “see” some spatial structure, due to migration, failures, differences in communication latency, and other phenomena ...

Proof Systems for Spatial Logics

A Proof System for Spatial Logic

- We define a labeled sequent calculus where labels denote π-calculus processes and accessibility is reduction:

$$\langle S \rangle\ u_1 : \mathcal{A}_1, \ldots, u_n : \mathcal{A}_n \vdash v_1 : \mathcal{B}_1, \ldots, v_m : \mathcal{B}_m$$

$\mathcal{A}_i$, $\mathcal{B}_j$ are formulas

u_i, v_j, labels are indexes, elements of

the term π-algebra $\mathbf{P} = \langle N, I, \mathbf{0}, |, \nu, \leftrightarrow_N, \leftrightarrow_I \rangle$ over process variables X, where N are name terms, and I are process terms.

S is a finite set of constraints, describing the "current world"

Constraints are either:

Equations $u = v$ between indexes (to handle spatial structure)

Distinctions $n \# m$ (to handle freshness)

Reductions $u \to v$ (to handle dynamics)

A Proof System for Spatial Logic

- Closure axioms for constraint sets, *e.g.*,

$$(\nu n)(\,u \mid (\nu n)v\,) \doteq_S ((\nu n)u) \mid (\nu n)v \qquad n \# p,\ m \# p \Rightarrow (m \leftrightarrow n)p \doteq_S p$$

- Propositional Rules, *e.g.*:

(∧ L)
$$\frac{\langle S\rangle\ \Gamma,\ u{:}\mathcal{A},\ u{:}\mathcal{B} \vdash \Delta}{\langle S\rangle\ \Gamma,\ u{:}\mathcal{A} \wedge \mathcal{B} \vdash \Delta}$$

(∧ R)
$$\frac{\langle S\rangle\ \Gamma \vdash u{:}\mathcal{A},\ \Delta \quad \langle S\rangle\ \Gamma \vdash u{:}\mathcal{B},\ \Delta}{\langle S\rangle\ \Gamma \vdash u{:}\mathcal{A} \wedge \mathcal{B},\ \Delta}$$

- Spatial Rules, *e.g.*:

(| L) $\mathcal{X},\mathcal{Y}$ *not free in the conclusion*
$$\frac{\langle S,\ u \doteq \mathcal{X} \mid \mathcal{Y}\rangle\ \Gamma,\ \mathcal{X}:\mathcal{A},\ \mathcal{Y}:\mathcal{B} \vdash \Delta}{\langle S\rangle\ \Gamma,\ u:\mathcal{A} \mid \mathcal{B} \vdash \Delta}$$

(| R)
$$\frac{\langle S\rangle\ \Gamma \vdash v{:}\mathcal{A},\ \Delta\ \langle S\rangle\ \Gamma \vdash t{:}\mathcal{B},\ \Delta \quad u \doteq_S v \mid t}{\langle S\rangle\ \Gamma \vdash u{:}\mathcal{A} \mid \mathcal{B},\ \Delta}$$

- World Rules, *e.g.*:

$$\frac{\langle\ S,\ u \doteq \mathbf{0}\rangle\ \Gamma \vdash \Delta \quad u \mid v \doteq_S \mathbf{0}}{\langle S\rangle\ \Gamma \vdash \Delta}$$

- Freshness Rules, *e.g.*:

(#) $\mathcal{Y}, x$ *not free in the conclusion*
$$\frac{\langle S,\ \mathbf{x} \# \mathbf{N},\ u \doteq (\nu \mathbf{x})\mathcal{Y}\rangle\ \Gamma \vdash \Delta}{\langle S\rangle\ \Gamma \vdash \Delta}$$

A Simple Proof

(0 R)

$$\frac{u \doteq_S \mathbf{0}}{\langle\, S \,\rangle\, \Gamma \vdash u : \mathbf{0},\, \Delta}$$

(| R)

$$\frac{\langle\, S \,\rangle\, \Gamma \vdash v : \mathcal{A},\, \Delta \quad \langle\, S \,\rangle\, \Gamma \vdash t : \mathcal{B},\, \Delta \quad u \doteq_S v|t}{\langle\, S \,\rangle\, \Gamma \vdash u : \mathcal{A} \mid \mathcal{B},\, \Delta}$$

(0 L)

$$\frac{\langle S,\, u \doteq \mathbf{0}\rangle\, \Gamma \vdash \Delta}{\langle S\rangle\, \Gamma,\, u : \mathbf{0} \vdash \Delta}$$

(| L) $\mathcal{X}, \mathcal{Y}$ *not free in the conclusion*

$$\frac{\langle\, S,\, u \doteq \mathcal{X} | \mathcal{Y}\rangle\, \Gamma,\, \mathcal{X} : \mathcal{A},\, \mathcal{Y} : \mathcal{B} \vdash \Delta}{\langle\, S \,\rangle\, \Gamma,\, u : \mathcal{A} \mid \mathcal{B} \vdash \Delta}$$

5 $\langle\, \mathcal{Z} \doteq \mathcal{X}|\mathcal{Y},\, \mathcal{Z} \doteq 0,\, \mathcal{X} \doteq 0 \,\rangle\, \mathcal{X} : \mathcal{A},\, \mathcal{Y} : \mathcal{B} \vdash \mathcal{Z} : \mathcal{A}$ — (Id) since $z = x$

4 $\langle\, \mathcal{Z} \doteq \mathcal{X}|\mathcal{Y},\, \mathcal{Z} \doteq 0 \,\rangle\, \mathcal{X} : \mathcal{A},\, \mathcal{Y} : \mathcal{B} \vdash \mathcal{Z} : \mathcal{A}$ — 5, (S | 0) since $x \mid y = 0$

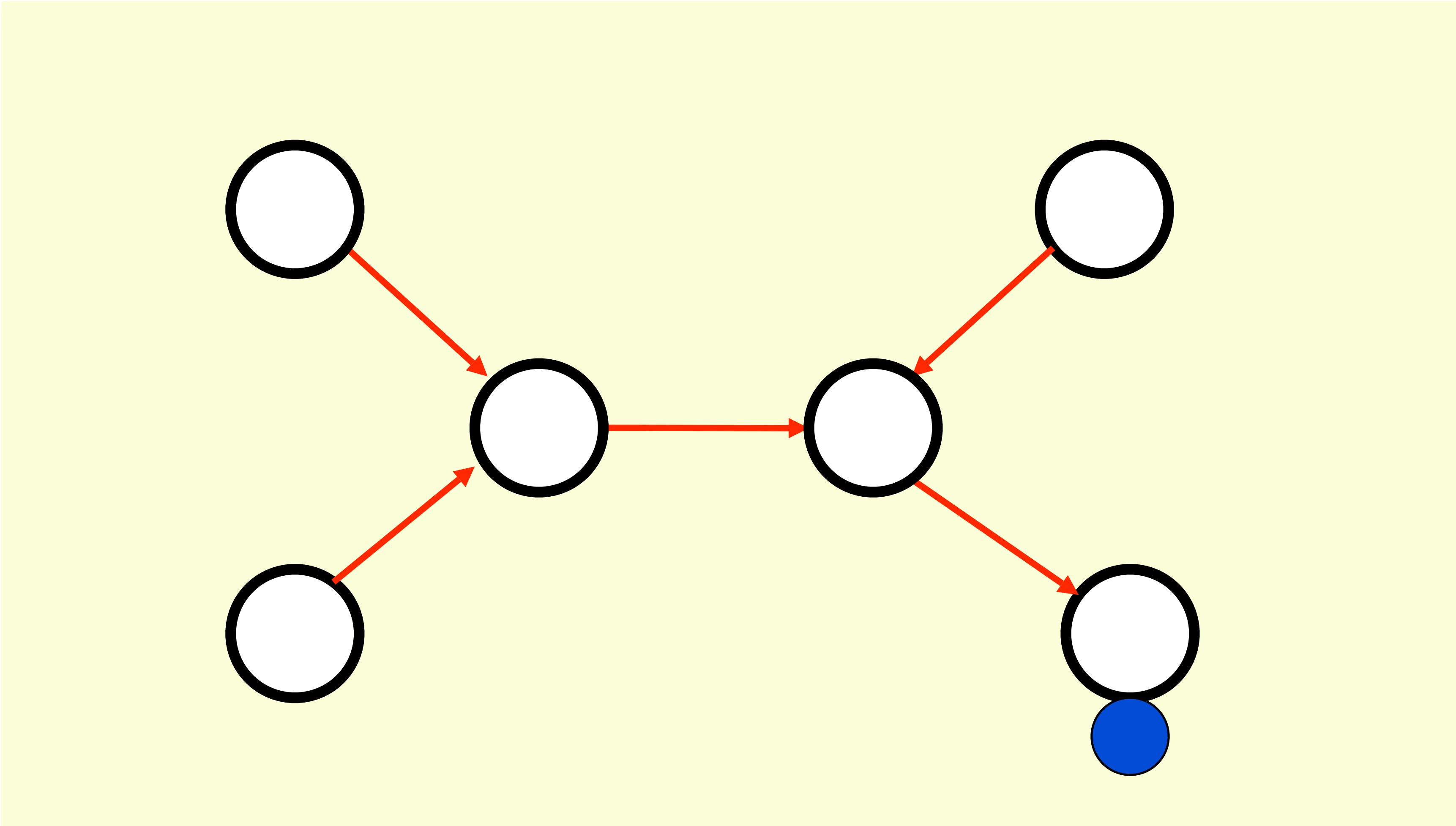
3 $\langle\, \mathcal{Z} \doteq \mathcal{X}|\mathcal{Y}\rangle\, \mathcal{X} : \mathcal{A},\, \mathcal{Y} : \mathcal{B},\, \mathcal{Z} : \mathbf{0} \vdash \mathcal{Z} : \mathcal{A}$ — 4, (0 L)

2 $\langle\,\rangle\, \mathcal{Z} : \mathcal{A} \mid \mathcal{B},\, \mathcal{Z} : \mathbf{0} \vdash \mathcal{Z} : \mathcal{A}$ — 3, (| L)

1 $\langle\,\rangle\, \mathcal{Z} : (\mathcal{A} \mid \mathcal{B}) \wedge \mathbf{0} \vdash \mathcal{Z} : \mathcal{A}$ — 2, (∧ L)

Reasoning with Spatial Logic

Spatial - Behavioral Properties



- "It is always possible for any site to eventually acquire exclusive access to the resource"

Specifications for Directory Nodes

$$\mathrm{idle}(f, m, l) \Leftrightarrow \mathbf{1} \wedge \forall\alpha.[\,\alpha\,] \; \exists\, a, n.(\; \alpha = f?(a, n) \wedge (\, \mathrm{idle}(f, m, n)) \mid l!(a, f)\,) \vee (\alpha = \tau \wedge (\,\mathrm{twaiter}(f, m)) \mid l!(m, f)\,)$$

$$\mathrm{towner}(f, m) \Leftrightarrow \mathbf{1} \wedge \forall\alpha.[\,\alpha\,] \; \exists\, a, n.\, (\alpha = f?(a, n) \wedge \mathrm{owner}(f, m, n, a))$$

$$\mathrm{owner}(f, m, l, q) \Leftrightarrow \ldots$$

$$\mathrm{twaiter}(f, m) \Leftrightarrow \ldots$$

$$\mathrm{waiter}(f, m, l, q) \Leftrightarrow \ldots$$

$$\mathrm{node}(\,f\,) \Leftrightarrow \exists\, m, l, q.\; \mathrm{idle}(f, m, l) \vee \mathrm{twaiter}(f, m) \vee \mathrm{towner}(f, m) \vee \mathrm{owner}(f, m, l, q) \vee \mathrm{waiter}(f, m, l, q)$$

Specifications for the Directory

$$\begin{aligned}
\text{BootState} \quad &\triangleq \quad \exists\, r, u.(\ \ \text{towner}(f, u) \mid \\
&\qquad\qquad (\exists\, f, m, l.\ \text{idle}(f, m, l\,))^{*}\ ) \\
&\qquad\qquad \wedge\ \forall n.\ (?\text{node}(n) \Rightarrow\ ?\text{Path}(n, r)) \\
\text{System} \quad &\triangleq \quad (\exists\, n.\ \text{node}(n) \vee \text{Msg})^{*}
\end{aligned}$$

$$\text{Link}(a, b) \triangleq \ \exists\, c.\ b!(c, a) \vee \text{LinkedNode}(a, b))$$

$$\text{Path}(a, b) \Leftrightarrow\ (a = b) \vee \exists\, c.(\ \text{Link}(a, c) \vee \text{Link}(c, a) \mid \text{Path}(c, b)\ )$$

$$\text{UniquePath} \quad \triangleq \quad \neg\ \ \exists\, a, b.\ (a \neq b) \wedge (?\text{Path}(a, b) \mid\ ?\text{Path}(a, b))$$

N.B.: $\text{UniquePath} \vdash \text{NoCycle}$

$$\begin{aligned}
\text{Ok} \quad &\triangleq \quad \text{UniquePath} \\
\text{NeverLoops} \quad &\triangleq \quad \text{always}(\text{Ok})
\end{aligned}$$

Proving a Safety Property

- Plan to show: Ok $\Rightarrow \Box$ Ok

  Assume: Ok $\wedge$ System

  Then prove: ($\Diamond \neg$ Ok) $\Rightarrow$ False

NeverLoops $\triangleq$ always(Ok)

Ok $\triangleq$ UniquePath

- Assume: $\Diamond \neg$ Ok

  Use the rule: $\Diamond \mathcal{A} \vdash \exists \mathcal{X}, m. (m \mid \mathcal{X} \mid \mathbf{1} \;\wedge \langle m \rangle (\mathcal{X} \rhd \mathcal{A}))$

  $\exists \mathcal{X}, m. ((m \wedge \text{Msg}) \mid (\mathcal{X} \wedge \text{Ok}) \mid (\text{Node} \wedge \langle m \rangle (\mathcal{X} \rhd \neg \text{Ok})))$

- Then, proceed by case analysis on Node $\triangleq \exists\, n.\text{node}(n)$

  $\exists \mathcal{X}. \Diamond (\mathcal{X} \wedge \text{Ok} \mid (\mathit{Post} \wedge (\mathcal{X} \rhd \neg \text{Ok}))$

  Ok | *Post* $\Rightarrow$ Ok

  $\Diamond$ (Ok $\wedge \neg$ Ok)

  $\Diamond$ False

  False

Msg $\Rightarrow \exists f,\ m, l. f!(m, l) \vee \exists m.\, m!()$

Node $\Rightarrow [\ m\ ]$ *Post*

Proving a Safety Property

- Details of one case (towner(v, n)):

$\exists\, \mathcal{X}, m.\ (\, m \mid (\mathcal{X} \wedge \mathrm{Ok}) \mid (\mathrm{Node} \wedge \langle m \rangle (\mathcal{X} \triangleright \neg\, \mathrm{Ok})))$

$\exists\, \mathcal{X}, m, v, n.\ (\, m \mid (\mathcal{X} \wedge \mathrm{Ok}) \mid (\mathrm{towner}(v, n) \wedge \langle m \rangle (\mathcal{X} \triangleright \neg \mathrm{Ok})))$

$\exists\, m,v,u.(v!(m, u) \mid (\mathcal{X} \wedge \mathrm{Ok}) \mid (\mathrm{towner}(v, n) \wedge \langle v?(m, u) \rangle (\mathcal{X} \triangleright \neg \mathrm{Ok})))$

$\exists\, m,v,u.\ (v!(m, u) \mid (\mathcal{X} \wedge \mathrm{Ok} \wedge \neg \mathrm{Path}(v, u)) \mid (\mathrm{towner}(v, n) \wedge$

$\langle\ v?(m, u) \rangle (\mathcal{X} \triangleright \neg \mathrm{Ok})))$

$\Diamond\ (\mathcal{X} \wedge \mathrm{Ok} \wedge \neg \mathrm{Path}(v, u) \mid (\mathrm{owner}(v, n, u, m) \wedge (\mathcal{X} \triangleright \neg \mathrm{Ok}))$

$\mathrm{Ok} \wedge \neg\ \mathrm{Path}(a, b) \Rightarrow (\mathrm{Path}(a, b) \triangleright \mathrm{Ok})$ (Auxiliary Lemma)

$\mathrm{Ok} \wedge \neg\ \mathrm{Path}(v, u) \mid \mathrm{owner}(v, n, u, m) \Rightarrow \mathrm{Ok}$

$\Diamond\ (\mathrm{Ok} \wedge \neg\ \mathrm{Ok})$

$\Diamond$ False

False

A Proof Rule for Safety Properties

- A property A is distributed if the following entailments hold:

$$A \mid A \vdash A$$

$$A \vdash \mathrm{True} \blacktriangleright A$$

For example, A* is distributed for all A.

In our example, property UniquePath is distributed.

- Then, the following proof rule is sound for any distributed G:

$$\frac{\mathbf{S} \vdash (\bigvee_{\mathbf{i}} \mathbf{C}_{\mathbf{i}})^* \qquad \mathbf{C}_{\mathbf{i}} \vdash \forall\alpha.\,[\alpha]\ \bigvee_{\mathbf{j}} (\alpha = m_{\mathrm{j}} \wedge \mathrm{P}_{\mathrm{j}}) \qquad \bigwedge_{\mathbf{j}} \mathrm{G} \Rightarrow (m_{\mathrm{j}} \blacktriangleright (\mathrm{P}_{\mathrm{j}} \triangleright \mathrm{G}))}{\mathbf{S} \wedge \mathrm{G} \vdash \Box \mathrm{G}}$$

A Spatial Type System for Services

Some Key Features of Distributed Services

- **Distributed Services**
  - Also known as distributed objects, but ...
  - On-the-fly system composition
  - Services depend (call) on other (remote) services
  - Tasks must be properly scheduled (workflow)
  - Services may be dynamically bound and (ref) passed around
- **Procedural Abstraction for Tasks**
  - Using distributed remote procedure call mechanisms
  - Control flow involves distributed (long term) transactions
- **Coordination Abstractions**
  - Parallel Composition of Tasks
    - Tasks are independent when never get to compete for resources
    - Independent tasks appear to run “simultaneously” (no interference)
    - This is the default behavior of the “global computer”
  - Sequential Composition of Tasks
    - Causality, data flow, and resource competition leads to sequentiality

Some Key Features of Distributed Services

- **Resources as disciplined objects**

  Resources are (at least partially) unshareable services/objects

  E.g., a file, a session, a key, a physical device, …

  Must be used according to a strict discipline / protocol (otherwise faults may occur)

- **Control of Resources**

  Resources may be passed around, buffered in pools, etc, ...

  In principle, at any given moment the sets of resources usable by each one of the ongoing tasks should be kept disjoint (*cf*., the separation principle).

  However, resource sharing policies may be of fine granularity (think of “multiple readers-unique writer” as a special case)

- **Spatial-Behavioral Types**

  We develop a core programming language and a compositional type system to discipline interactions and resource usage on distributed services systems, inspired by spatial logic.

A General Type Structure for Services

- $U \mid V$

  Independent Tasks (Spatial Decomposition)

- $U \& V$

  Optional Tasks

- U ; V

  Sequential Tasks

- T°

  Owned Task

- U ▷ V

  Guarantee (Spatial Compositionality)

- **Spatial Logic Semantics of Types**

  $P \models T$ (logical satisfaction, compositionally defined)

  Types denote “properties of processes”, not “processes”

  The semantics of types entails the intended safety properties

Composition with Spatial Types

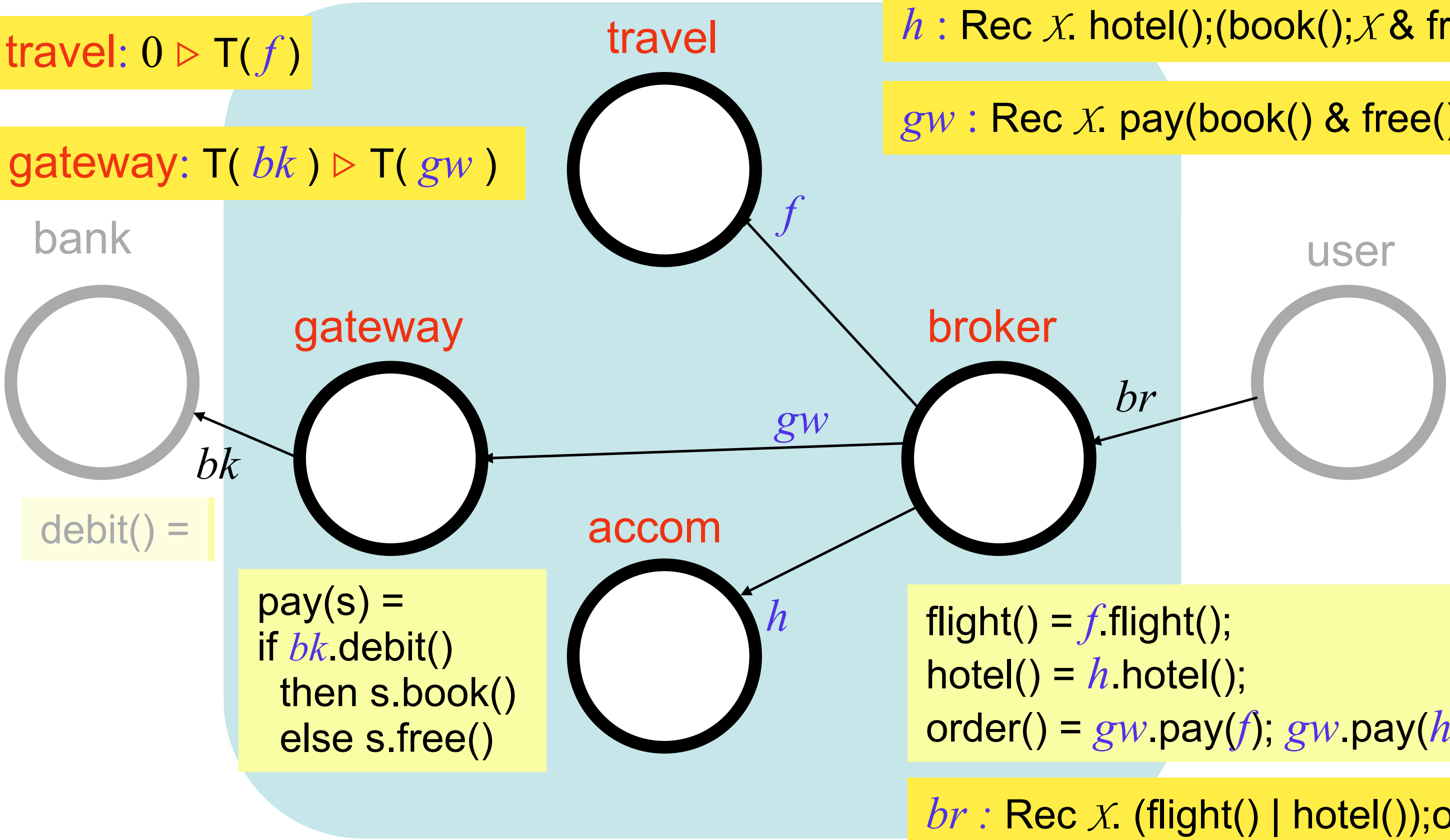
f : Rec X. flight();(book();X & free();X)

h : Rec X. hotel();(book();X & free();X)

gw : Rec X. pay(book() & free());X

travel: $0 \triangleright$ T(f)

gateway: T(bk) $\triangleright$ T(gw)



pay(s) =
if bk.debit()
  then s.book()
  else s.free()

flight() = f.flight();
hotel() = h.hotel();
order() = gw.pay(f); gw.pay(h);

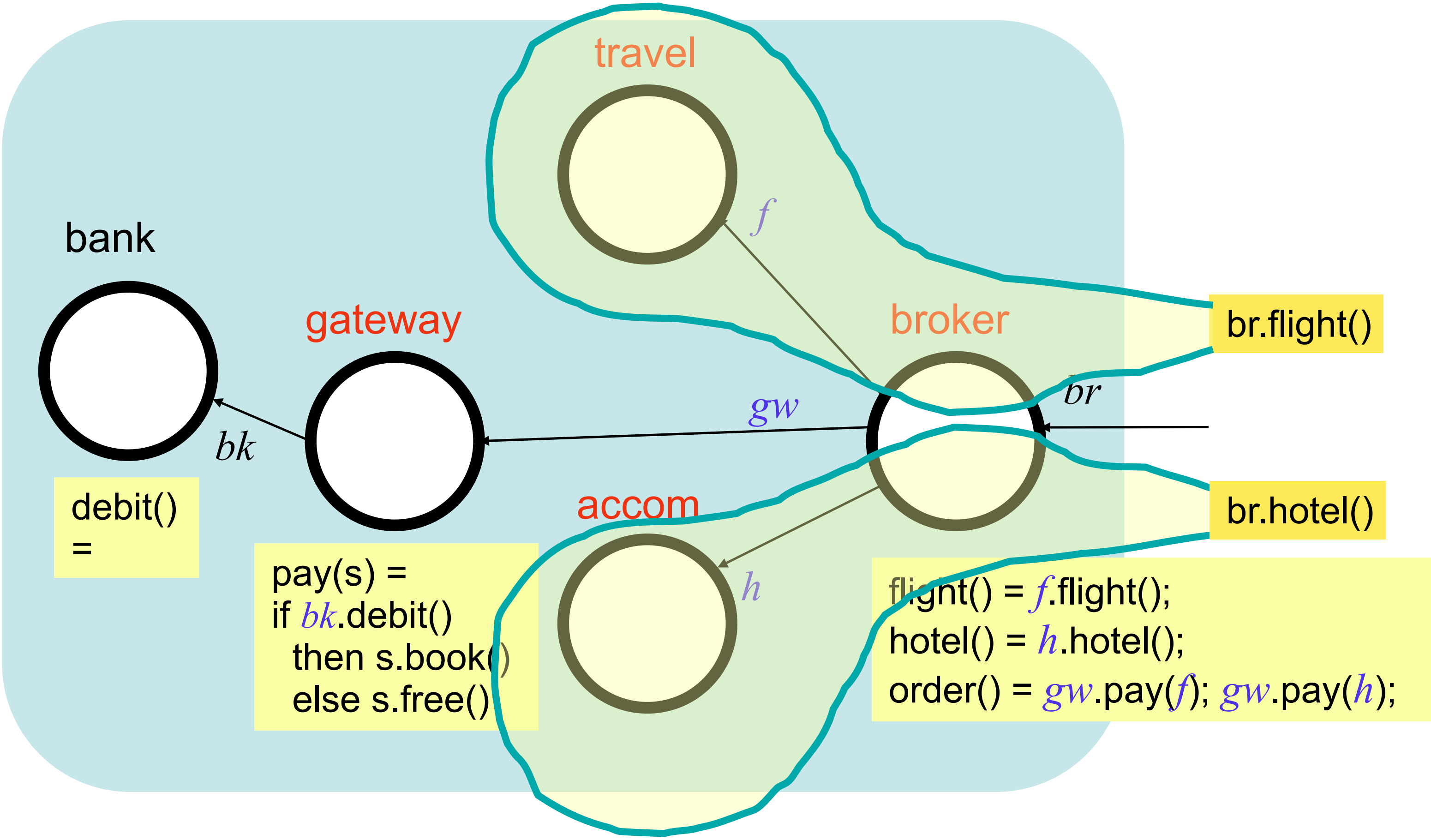
br : Rec X. (flight() | hotel());order();X

BackSys: T(bk) $\triangleright$ T(gw) | T(f) | T(h)

broker: (T(gw) | T(f) | T(h)) $\triangleright$ T(br)

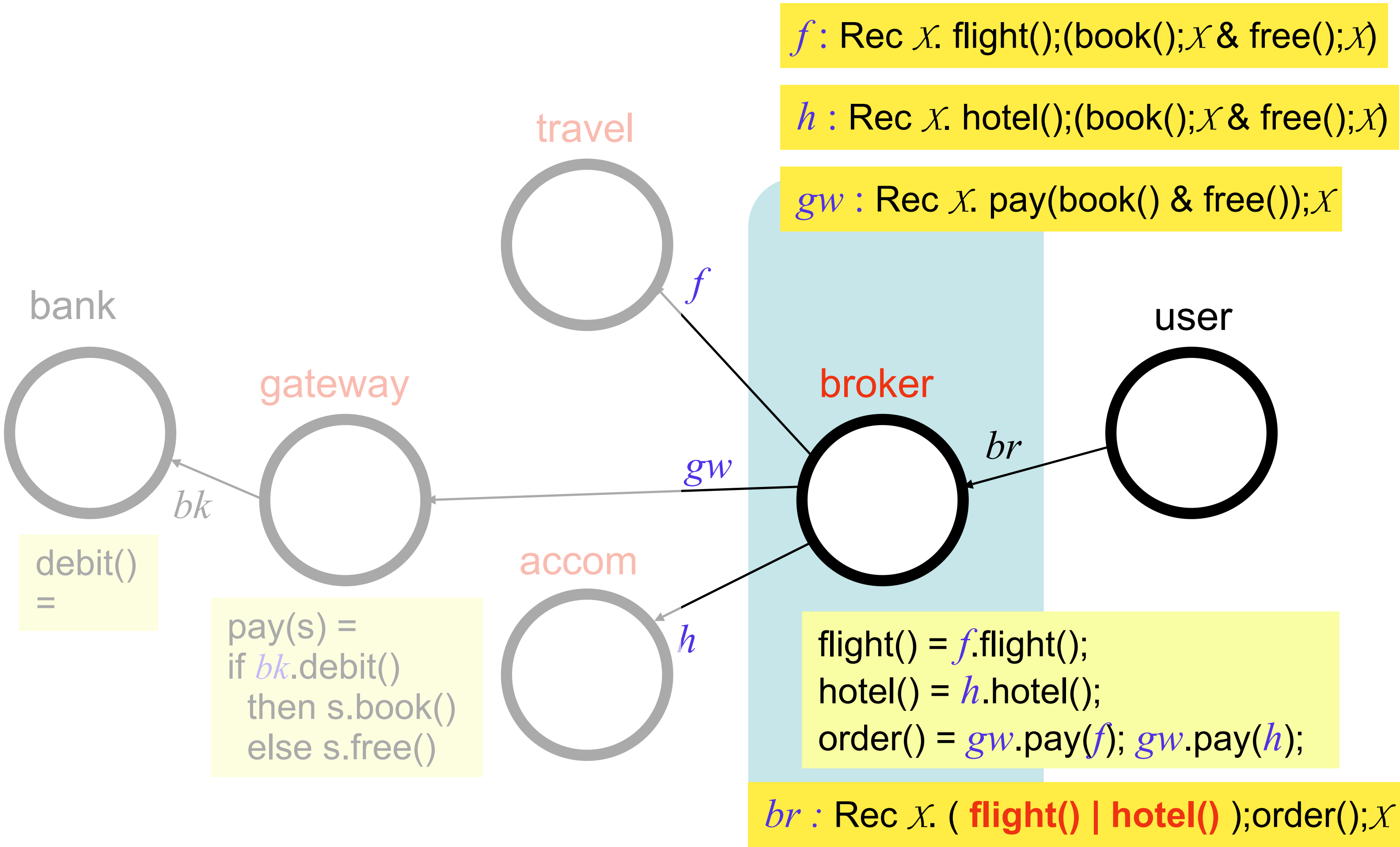
TravelSvc: T(bk) $\triangleright$ T(br)

Spatial Decomposition



Each of the service calls *br*.hotel() and *br*.flight() is handled by (spatially) separated parts of the system.

Reflecting Spatial Decomposition on Types



The semantics of the (*U* | *V*) typing implies that the services *br*.hotel() and *br*.flight() may be (safely) used concurrently.

Composing BackSys and broker

bank: $0 \triangleright \mathrm{T}(\, b \,)$

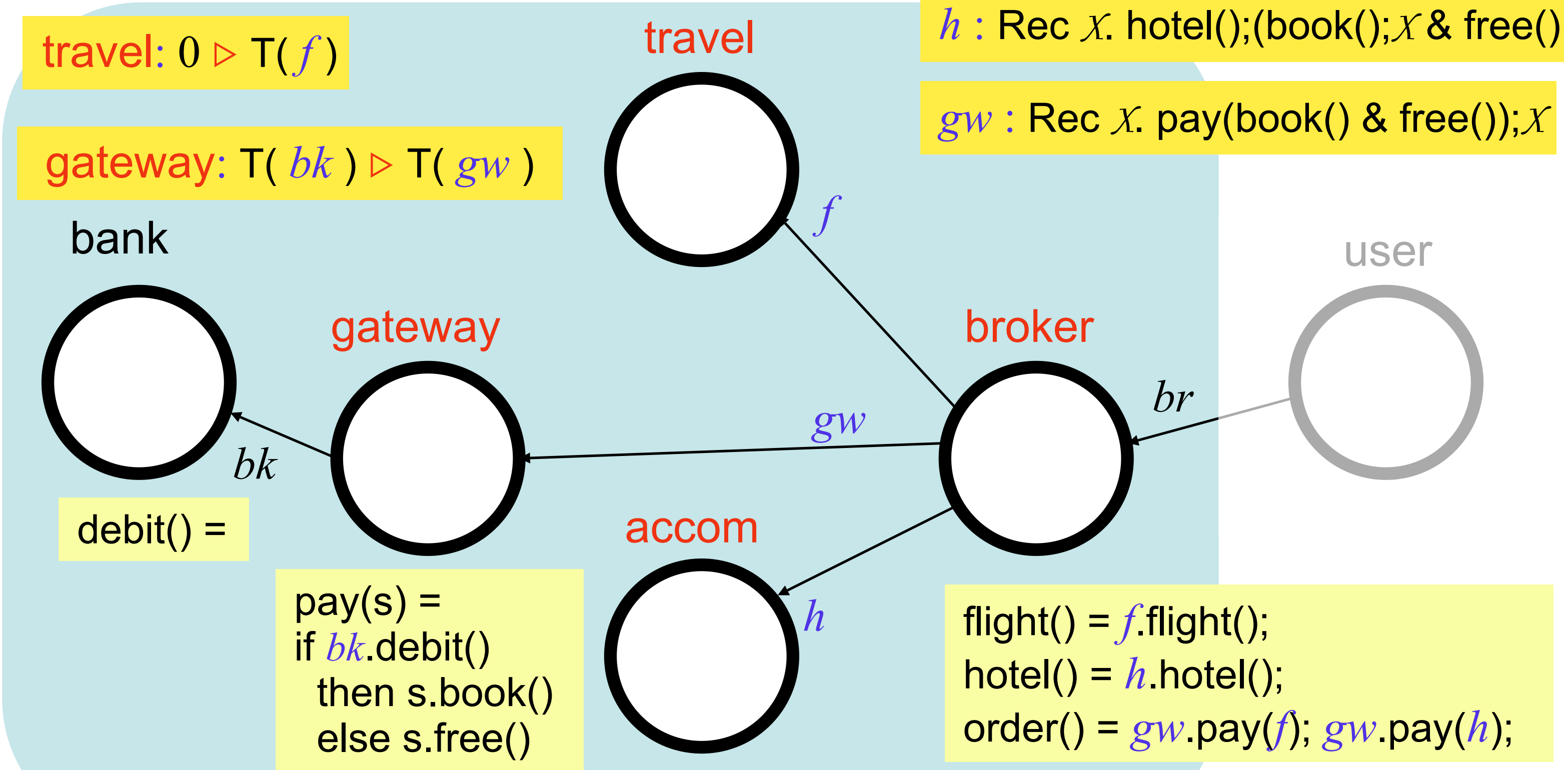
travel: $0 \triangleright \mathrm{T}(\, f \,)$

gateway: $\mathrm{T}(\, bk \,) \triangleright \mathrm{T}(\, gw \,)$

f : Rec X. flight();(book();X & free();X)

h : Rec X. hotel();(book();X & free();X)

gw : Rec X. pay(book() & free());X



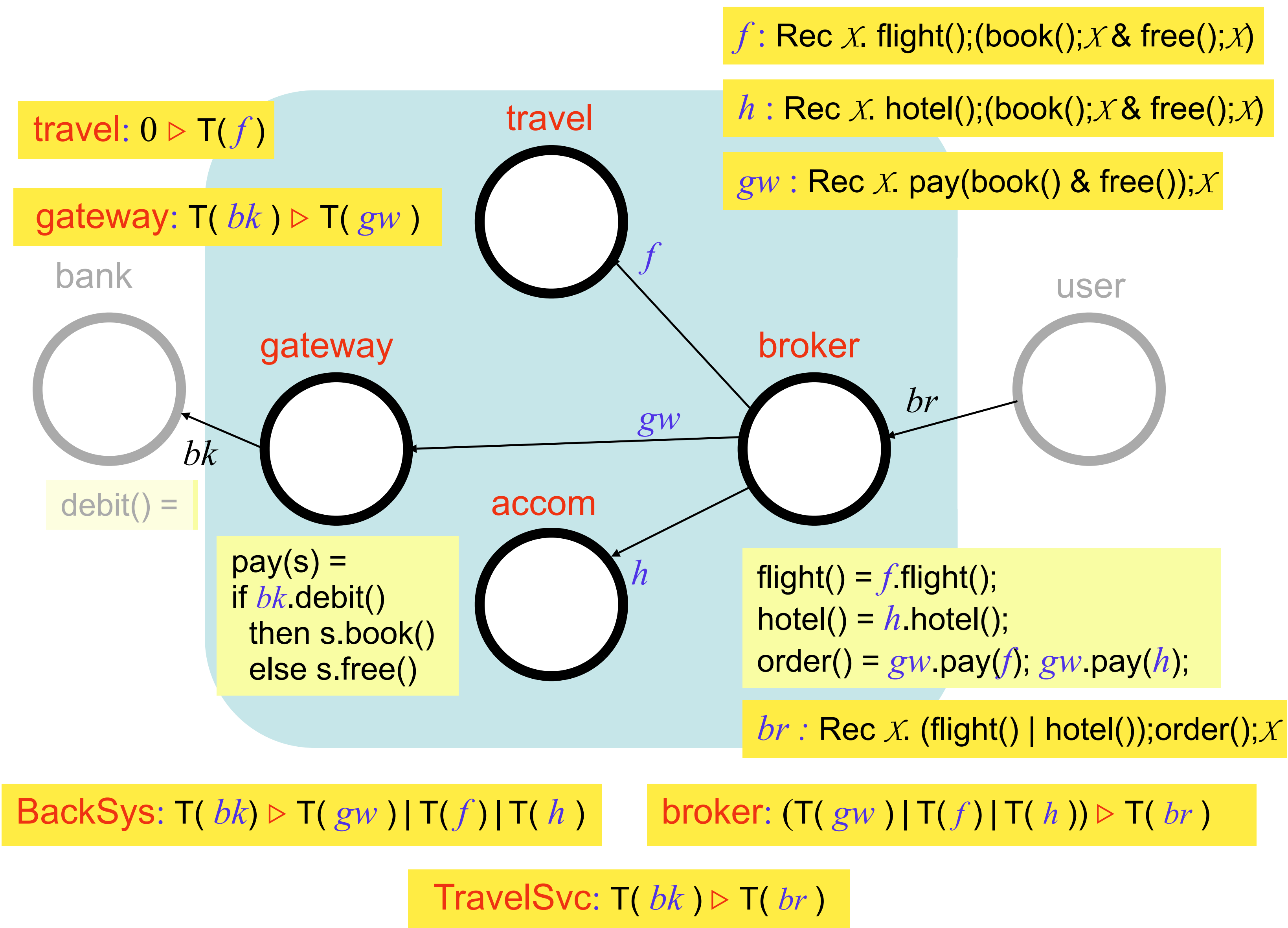
br : Rec X. (flight() | hotel());order();X

BackSys: $0 \triangleright \mathrm{T}(\, gw \,) \mid \mathrm{T}(\, f \,) \mid \mathrm{T}(\, h \,)$

broker: $(\mathrm{T}(\, gw \,) \mid \mathrm{T}(\, f \,) \mid \mathrm{T}(\, h \,)) \triangleright \mathrm{T}(\, br \,)$

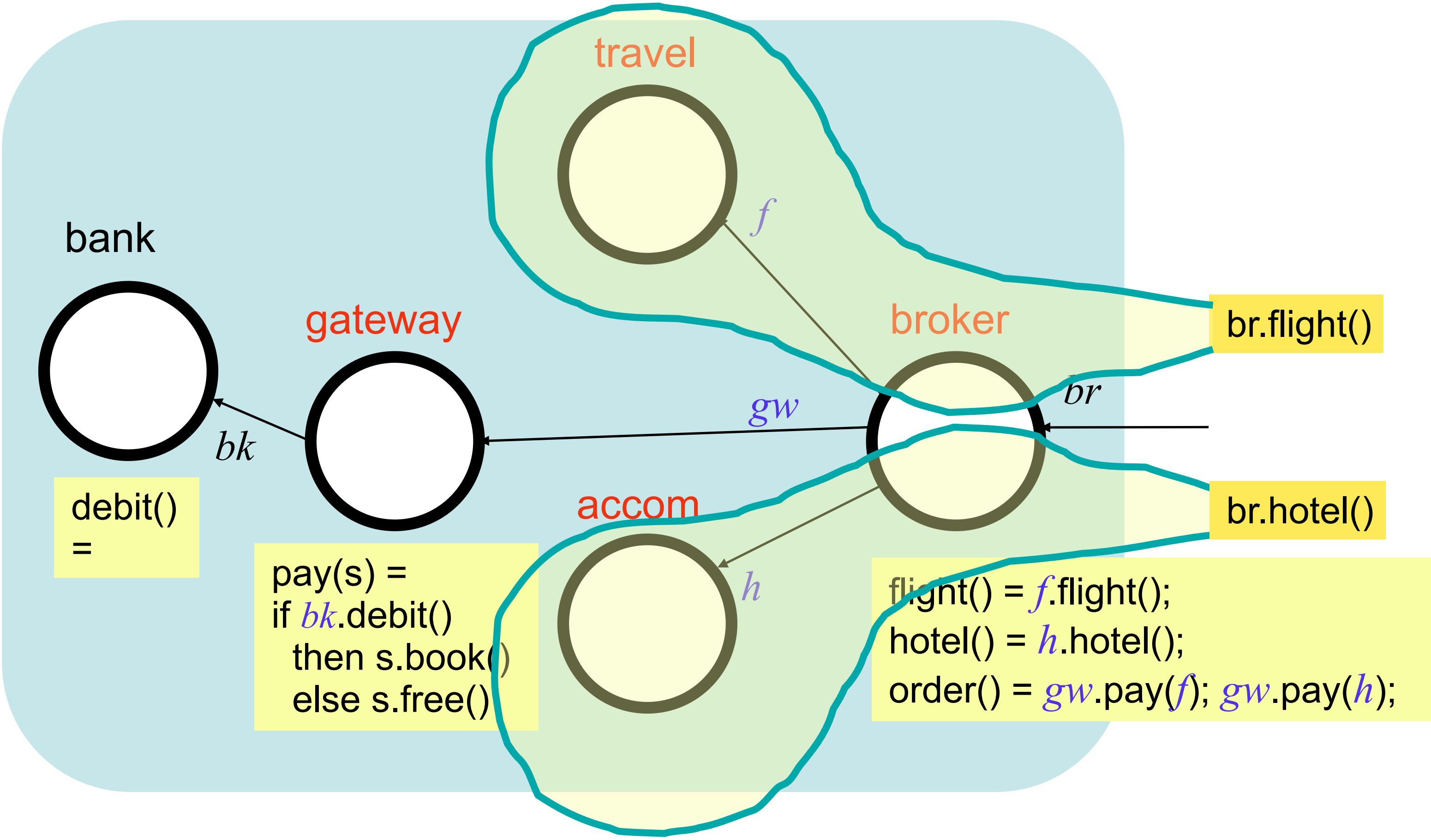
TravelSys: $0 \triangleright \mathrm{T}(\, br \,)$

Another Decomposition



Intended Safety Properties of Typing

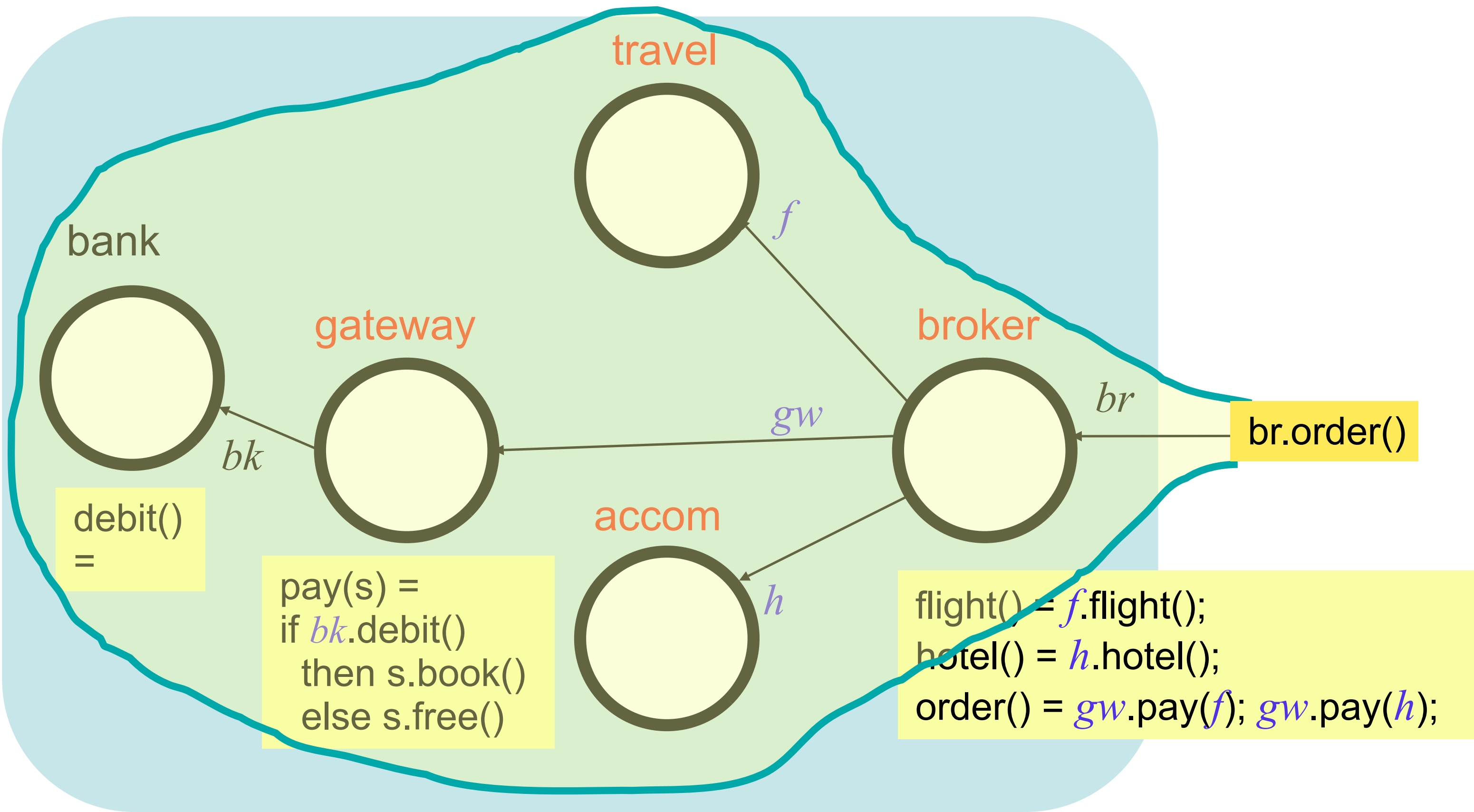
Flow ≜ Available ∧ ¬Race ∧ Unique   Safe ≜ ⊡Flow



The subsystem responsible for the distributed execution of a given service call should always satisfy the Flow invariant.

Intended Safety Properties of Typing

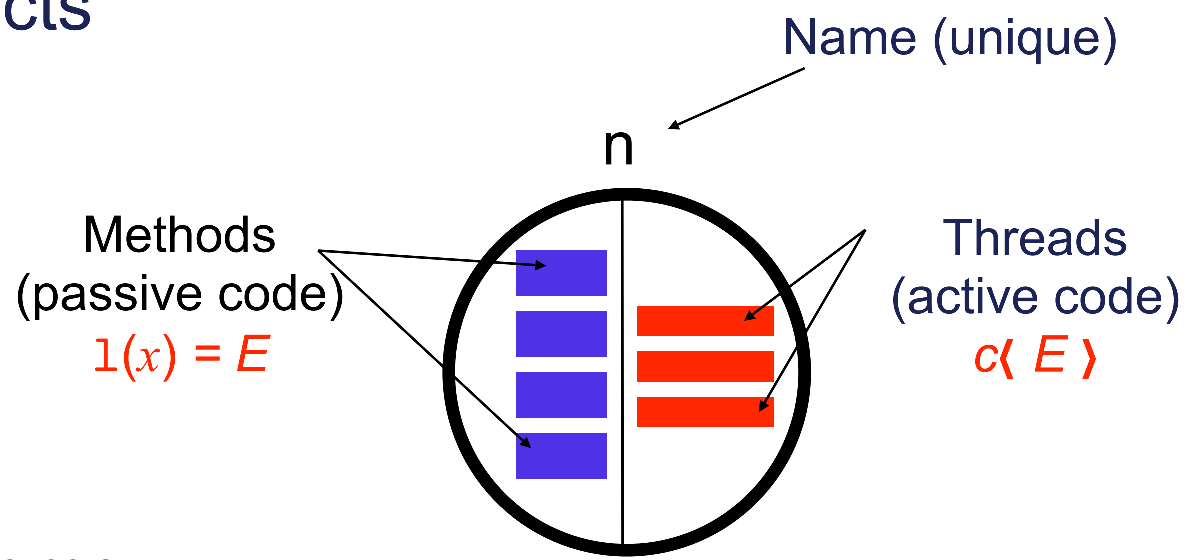
Flow ≜ Available ∧ ¬Race ∧ Unique   Safe ≜ ⊡Flow



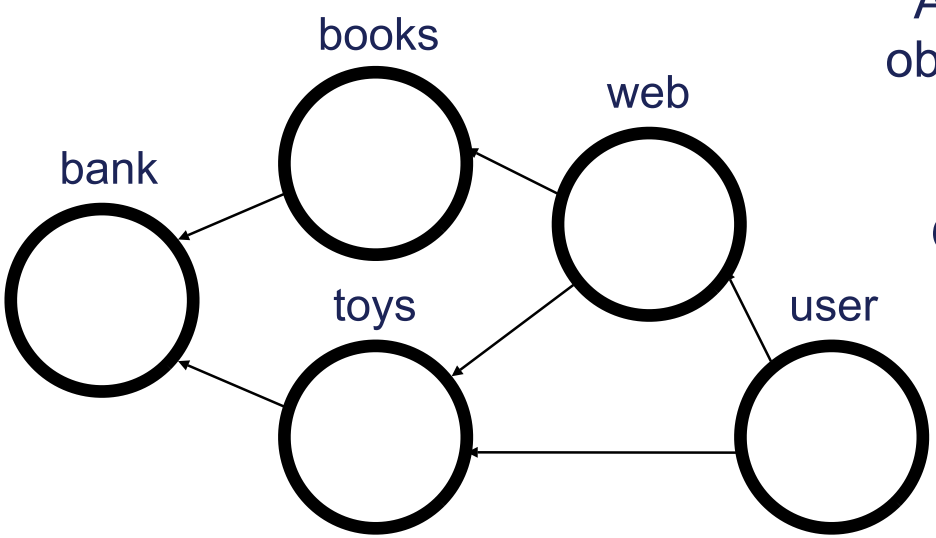
The subsystem responsible for the distributed execution of a given service call should always satisfy the Flow invariant.

The Object Model

Objects



Systems



A system is composed by several objects, each one possibly running several threads concurrently

Objects may call methods of any other objects they know about

Object names may be passed around (name mobility)

The Process Calculus

$n, m, p \in$ *Names*	
$u, v \in$ *Val* ::=	**Values**
stop	*Primitive*
n	*Name*

$x, y, z \in$ *Variables*	
$E, F \in$ *Expr* ::=	**Expressions**
x	*Identifier*
v	*Value*
$n.\mathtt{l}(v)$	*Call*
$n.c()$	*Back*
let $x = E$ **in** F	*Composition*
new $[\mathcal{M}]$	*Instantiation*

$\mathtt{l} \in$ *Labels*	
$\mathcal{M} \in$ *Meth* ::=	**Methods**
0	*Void*
$\mathtt{l}(x) = E$	*Method*
$\mathcal{M} \mid \mathcal{M}$	*Methods*

$T, S \in$ *Thr* ::=	**Threads**
0	*Void*
$c\langle E \rangle$	*Thread*
$T \mid S$	*Threads*

$P, Q, R \in$ *Proc* ::=	**Processes**
0	*Void*
$n[\mathcal{M} ; T]$	*Object*
$P \mid Q$	*Objects*

N.B.:

- In any object $n[\mathcal{M} ; T]$

Methods in $\mathcal{M}$ have **distinct labels.**

Threads in T have **distinct names.**

- In any process P

Objects in P have **distinct names.**

Types

$\mathcal{X}, \mathcal{Y}, \mathcal{Z} \in$ *Type Variables*

T,U,V $\in$ *Type* ::= **Types**

nil	*Stop*
`l`(*U*)V	*Method*
c(m:*U*)V	*Thread*
U | V	*Spatial*
U ; V	*Sequential*
U & V	*Conjunction*
U^{o}	*Owned*
$\mathcal{X}$	*Variable*
Rec $\mathcal{X}$.*T*	*Recursion*

Atomic Typing Assertion.

n : *T* (object n has type *T*)

Typing Contexts (A, B).

$n_1 : T_1, \ldots, n_k : T_k$

N.B. write $A(n_1) = T_1$

when A is $n_1 : T_1, \ldots, n_k : T_k$

Typing Judgment (Expressions).

$E :: A \triangleright B\ [T]$

Typing Judgment (Processes).

$P :: A \triangleright B$

A valid expression judgment.

A $\triangleq$	*n*:	`fly();fr(),`
	h:	`hot();bk()&fr(),`
	g:	`do(bk())`*T*
B $\triangleq$	*n*:	`fr(),`
	h:	**nil**,
	g:	**nil**
E $\triangleq$	(*n*.`fly`() | *h*.`hot`()); g.do(*h*)	

$E :: A \triangleright B\ [T]$

Subtyping

Congruences.

$U <: U', V <: V' \Rightarrow U \mid V <: U' \mid V'$

$U <: U', V <: V' \Rightarrow U ; V <: U' ; V'$

etc...

Conjunction.

$U \& V <: U$

$U \& V <: V$

$T <: U, T <: V \Rightarrow T <: U \& V$

Composition.

$U \mid \mathbf{nil} <:> U$

$U \mid V <:> V \mid U$

$(U \mid V) \mid T <:> U \mid (V \mid T)$

Sequential.

$U ; \mathbf{nil} <:> U$

$U \mid V <: U ; V$

$\mathbf{nil} ; U <:> U$

$(U ; V) ; T <:> U ; (V ; T)$

$(U ; U') \mid (V ; V') <: (U \mid V) ; (U' \mid V')$

Owned.

$U^{o} <: U$

$\mathbf{nil} <: \mathbf{nil}^{o}$

$U^{o} <: \mathbf{nil}$

$U^{o} <: U^{oo}$

$(U \mid V)^{o} <: U^{o} \mid V^{o}$

$(U ; V)^{o} <: U ; V^{o}$

$U^{o} ; V <: U^{o} \mid V$

Contexts.

$V <: U \Rightarrow n{:}V <: n{:}U'$

Soundness of Typing

- There is an embedding $\lceil \ \rceil$ of our types into a spatial logic such that:

Soundness of Typing.

Let $P :: A \triangleright B$. Then $P \vDash \lceil A \rceil \triangleright \lceil B \rceil$.

Let $P :: \triangleright A$. Then $P \vDash \lceil A \rceil$.

- In particular, if $P :: \triangleright A$ is derivable then $P \vDash \mathrm{Safe}$ holds
- The soundness proof is driven by semantical reasoning (think of $A \triangleright B$ as a logical relation).
- It is very natural and modular (we look once at each rule).
- The type system is indeed a proof system (in the standard sense) for satisfaction ($P \vDash A$) w.r.t. the underlying logic.

State and Resource Control (Example)

R ≜ use()*
PoolType ≜ !(free(R°) & alloc()R)
pool : PoolType

```
server [
  init() = s!(stop);
  open() =
      let x = pool.alloc() in s!(x)
  use() =
      let x = s? in ( x.use() ; s!(x) )
  close() =
      let x = s?
      in ( pool.free(x); s!(stop); )
]
```

ResType ≜ use()*
SrvType ≜ init(); (open(); use()*; close())*
server[...] :: pool : PoolType ▷ server : Srvtype

Sharing and Resource Control (Example)

List $\triangleq$ add(Elt)*
BagType $\triangleq$!(pick()List & drop(List^{o}))
r : BagType

```
server [

   *login() =
        let l = new ListO in
           new [
              buy(i) = l.add(i)
              quit() = r.drop(l)
           ]

  dump() = r.pick()

]
```

Session $\triangleq$ Rec X. (buy(Elt);X & quit())
SrvType $\triangleq$ (! login()Session) | (dump()List)*
server[...] :: r : BagType $\triangleright$ server : Srvtype

Road Map

- **Specifications**
  - Operational vs. Logical Specifications
  - Logics (review)
  - Program Logics
  - Modal Logics
- **Logics for Concurrency and Distribution**
  - Hennessy-Milner Logics
  - μ-Calculi
  - Spatial Logics
- **Verification Techniques**
  - Model Checking
  - Proof Systems
  - Type Systems
- **References and (lots of) Further Reading**
  - Check the Spatial Logic Model Cheker web site for an up to date annotated bibliography on logics for concurrency and distribution

Assignments