Part 4
Spatial Logics
Luca Cardelli
Andy Gordon \ Luis Caires
Properties of Secure Mobile Computation

• We would like to express properties of unique, private, hidden, and secret names:
  • “The applet is placed in a private sandbox.”
  • “The key exchange happens in a secret location.”
  • “A shared private key is established between two locations.”
  • “A fresh nonce is generated and transmitted.”

• Crucial to expressing this kind of properties is devising new logical quantifiers for fresh and hidden entities:
  • “There is a fresh (never used before) name such that …”
  • “There is a hidden (unnamable) location such that …”
  • N.B.: standard quantifiers are problematic. “There exists a sandbox containing the applet” is rather different from “There exists a fresh sandbox containing the applet” and from “There exists a hidden sandbox containing the applet”.


Approach

• Use a specification logic grounded in an operational model of mobility. (So soundness is not an issue.)

• Express properties of dynamically changing structures of locations.
  • Previous work [POPL’00].

• Express properties of hidden names. We split it into two logical tasks:
  • Quantify over fresh names. We adopt [Gabbay-Pitts].
  • Reveal hidden names, so we can talk about them.
  • Combine the two, to quantify over hidden locations.

  “There is a hidden location …” represented as:

  “There is a fresh name that can be used to reveal (mention) the hidden name of a location …”.
Our basic model of space is going to be *finite-depth edge-labeled unordered trees* (c.f. semistructured data, XML). For short: *spatial trees*, represented by a syntax of *spatial expressions*. Unbounded resources are represented by infinite branching:

```
Cambridge
  Eagle
    chair chair glass glass glass ...
    pint pint pint pint ...
```

```
Cambridge[Eagle[chair[0] | chair[0] | !glass[pint[0]]] | ...]
```
• These spatial expressions/trees are a subset of ambient expressions/trees, which can represent both the spatial and the temporal aspects of mobile computation.

• An ambient tree is a spatial tree with, possibly, threads at each node that can locally change the shape of the tree.

\[a[c[out \ a. \ in \ b. \ P]] \parallel b[0]\]
Spatial Logics

- We want to describe mobile behaviors. The ambient calculus provides an operational model, where spatial structures (agents, networks, etc.) are represented by nested locations.

- We also want to specify mobile behaviors. To this end, we devise an ambient logic that can talk about spatial structures.

**Processes**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(void)</td>
</tr>
<tr>
<td>(n[P])</td>
<td>(location)</td>
</tr>
<tr>
<td>(P \parallel Q)</td>
<td>(composition)</td>
</tr>
</tbody>
</table>

**Formulas**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(there is nothing here)</td>
</tr>
<tr>
<td>(n[A])</td>
<td>(there is one thing here)</td>
</tr>
<tr>
<td>(A \parallel B)</td>
<td>(there are two things here)</td>
</tr>
</tbody>
</table>

**Trees**

- \((void)\)
- \((location)\)
- \((composition)\)
Mobility

- Mobility is change of spatial structures over time.

\[ a[Q \mid c[\text{out } a. \text{ in } b. \ P]] \quad \mid b[R] \]
Mobility is change of spatial structures over time.

\[ a[Q] \quad | \quad c[\text{in } b. P] \quad | \quad b[R] \]
Mobility is change of spatial structures over time.
These often have the form:

- Right now, we have a spatial configuration, and later, we have another spatial configuration.
- E.g.: Right now, the agent is outside the firewall, …
Properties of Mobile Computation

These often have the form:

- Right now, we have a spatial configuration, and later, we have another spatial configuration.

- E.g.: Right now, the agent is outside the firewall, and later (after running an authentication protocol), the agent is inside the firewall.
In a modal logic, the truth of a formula is relative to a state (called a *world*).

- Temporal logic: current time.
- Program logic: current store contents.
- Epistemic logic: current knowledge. Etc.

In our case, the truth of a *space-time modal formula* is relative to the *here and now* of a process.

- The formula $n[0]$ is read:
  
  there is *here and now* an empty location called $n$

- The operator $n[A]$ is a single step in space (akin to the temporal *next*), which allows us talk about that place one step down into $n$.

- Other modal operators talk about undetermined times (in the *future*) and undetermined places (in the location tree).
Logical Formulas

\( \mathcal{A} \in \Phi ::= \) Formulas

- \( T \) is true
- \( \neg \mathcal{A} \) is negation
- \( \mathcal{A} \lor \mathcal{A}' \) is disjunction
- \( 0 \) is void
- \( \eta[\mathcal{A}] \) is location
- \( \mathcal{A} @ \eta \) is location adjunct
- \( \mathcal{A} | \mathcal{A}' \) is composition
- \( \mathcal{A} \triangleright \mathcal{A}' \) is composition adjunct
- \( \eta \Box \mathcal{A} \) is revelation
- \( \mathcal{A} \Box \eta \) is revelation adjunct
- \( \Diamond \mathcal{A} \) is somewhere modality
- \( \Diamond \mathcal{A} \) is sometime modality
- \( \forall x. \mathcal{A} \) is universal quantification over names

(\( \eta \) is a name \( n \) or a variable \( x \))
Simple Examples

1: \( p[T] \mid T \)
   there is a location \( p \) here (and possibly something else)

2: \( \Diamond 1 \)
   somewhere there is a location \( p \)

3: \( 2 \Rightarrow \Box 2 \)
   if there is a \( p \) somewhere, then forever there is a \( p \) somewhere

4: \( p[q[T] \mid T] \mid T \)
   there is a \( p \) with a child \( q \) here

5: \( \Diamond 4 \)
   somewhere there is a \( p \) with a child \( q \)
Examples

• \(\text{an } n \triangleq n[T] \mid T\)  \quad \text{there is now an } n \text{ here}
• \(\text{no } n \triangleq \neg \text{an } n\)  \quad \text{there is now no } n \text{ here}
• \(\text{one } n \triangleq n[T] \mid \text{no } n\)  \quad \text{there is now exactly one } n \text{ here}
• \(\mathcal{A} \forall \triangleq \neg (\neg \mathcal{A} \mid T)\)  \quad \text{everybody here satisfies } \mathcal{A}
• \((n[T] \implies n[\mathcal{A}]) \forall\)  \quad \text{every } n \text{ here satisfies } \mathcal{A}
• \(\forall (\forall (n[T] \implies n[\mathcal{A}])) \forall\)  \quad \text{every } n \text{ everywhere satisfies } \mathcal{A}
Satisfaction for Basic Trees

- $\models \mathbf{0}$

\[ n \quad \models n[\overline{A}] \quad \text{if} \quad \models \overline{A} \]

\[ \overline{P} \quad \overline{Q} \quad \models \overline{A} \lor \overline{B} \quad \text{if} \quad \models \overline{A} \quad \text{and} \quad \models \overline{B} \]

\[ \overline{P} \quad \models \overline{A} \land n \quad \text{if} \quad \models \overline{A} \]

\[ \overline{P} \quad \models \overline{A} \land \overline{B} \quad \text{if for all} \quad \models \overline{A} \quad \text{we have} \quad \models \overline{B} \]
Satisfaction for Somewhere/Sometime

N.B.: instead of $\Diamond \mathcal{A}$ and $\bigcirc \mathcal{A}$ we can use a “temporal next” operator $\circ \mathcal{A}$, along with the existing “spatial next” operator $n[\mathcal{A}]$, together with $\mu$-calculus style recursive formulas.
Satisfaction for Revelation

- Trees with hidden labels:

\[ P \overset{m}{=} P\{m\leftarrow n\} \]

\[ P \overset{n}{=} P \]

\[ P \overset{m}{=} P \]

\[ (n \neq m) \]

Etc.

\[ P \overset{n}{\models} n \otimes \mathcal{A} \quad \text{if} \quad P \overset{\mathcal{A}}{\models} \]

Not possible if \( n \) is free!

\[ P \overset{\mathcal{A} \otimes n}{\models} \quad \text{if} \quad P \overset{\mathcal{A}}{\models} \]
**Intended Model: Ambient Calculus**

\[ P \in \Pi ::= \begin{array}{l}
\begin{align*}
&\text{Processes} \\
&\quad (\forall n)P \quad \text{restriction} \\
&\quad 0 \quad \text{inactivity} \\
&\quad P \mid P' \quad \text{parallel} \\
&\quad M[P] \quad \text{ambient} \\
&\quad !P \quad \text{replication} \\
&\quad M.P \quad \text{exercise a capability} \\
&\quad (n).P \quad \text{input locally, bind to } n \\
&\quad \langle M \rangle \quad \text{output locally (async)}
\end{align*}
\end{array} \]

\[ M ::= \begin{array}{l}
\begin{align*}
&\text{Messages} \\
&\quad n \quad \text{name} \\
&\quad in M \quad \text{entry capability} \\
&\quad out M \quad \text{exit capability} \\
&\quad open M \quad \text{open capability} \\
&\quad \varepsilon \quad \text{empty path} \\
&\quad M.M' \quad \text{composite path}
\end{align*}
\end{array} \]

\[ n[] \triangleq n[0] \]

\[ M \triangleq M.0 \quad \text{(where appropriate)} \]
Reduction Semantics

- A structural congruence relation $P \equiv Q$:
  - On spatial expressions, $P \equiv Q$ iff $P$ and $Q$ denote the same tree. So, the syntax modulo $\equiv$ is a notation for spatial trees.
  - On full ambient expressions, $P \equiv Q$ if in addition the respective threads are “trivially equivalent”.
  - Prominent in the definition of the logic.

- A reduction relation $P \rightarrow^* Q$:
  - Defining the meaning of mobility and communication actions.
  - Closed up to structural congruence:
    $$P \equiv P', \ P' \rightarrow^* Q', \ Q' \equiv Q \ \Rightarrow \ P \rightarrow^* Q$$
Reduction

- Four basic reductions plus propagation, rearrangement (composition with structural congruence), and transitivity.

\[
\begin{align*}
n[\text{in } m. \ P \mid Q] \mid m[R] & \rightarrow m[n[P \mid Q] \mid R] \quad \text{(Red In)} \\
m[n[\text{out } m. \ P \mid Q] \mid R] & \rightarrow n[P \mid Q] \mid m[R] \quad \text{(Red Out)} \\
\text{open } m. \ P \mid m[Q] & \rightarrow P \mid Q \quad \text{(Red Open)} \\
(n).P \mid \langle M \rangle & \rightarrow P\{n\leftarrow M\} \quad \text{(Red Comm)}
\end{align*}
\]

\[
\begin{align*}
P \rightarrow Q & \Rightarrow (\forall n)P \rightarrow (\forall n)Q \quad \text{(Red Res)} \\
P \rightarrow Q & \Rightarrow n[P] \rightarrow n[Q] \quad \text{(Red Amb)} \\
P \rightarrow Q & \Rightarrow P \mid R \rightarrow Q \mid R \quad \text{(Red Par)}
\end{align*}
\]

\[
P' \equiv P, \ P \rightarrow Q, \ Q \equiv Q' \Rightarrow P' \rightarrow Q'
\]

\[\rightarrow^* \text{ is the reflexive-transitive closure of } \rightarrow\]
Structural Congruence

- Routine, but used heavily in the logic and semantics.

\[
\begin{align*}
P ≡ P & \quad \text{(Struct Refl)} \\
P ≡ Q & \implies Q ≡ P \quad \text{(Struct Symm)} \\
P ≡ Q, Q ≡ R & \implies P ≡ R \quad \text{(Struct Trans)} \\
P ≡ Q & \implies (∀n)P ≡ (∀n)Q \quad \text{(Struct Res)} \\
P ≡ Q & \implies P ∣ R ≡ Q ∣ R \quad \text{(Struct Par)} \\
P ≡ Q & \implies !P ≡ !Q \quad \text{(Struct Repl)} \\
P ≡ Q & \implies M[P] ≡ M[Q] \quad \text{(Struct Amb)} \\
P ≡ Q & \implies M.P ≡ M.Q \quad \text{(Struct Action)} \\
P ≡ Q & \implies (n).P ≡ (n).Q \quad \text{(Struct Input)} \\
ε.P ≡ P & \quad \text{(Struct ε)} \\
(M.M').P ≡ M.M'.P & \quad \text{(Struct .)}
\end{align*}
\]
\[
\begin{align*}
(\forall n)0 & \equiv 0 & \text{(Struct Res Zero)} \\
(\forall n)(\forall m)P & \equiv (\forall m)(\forall n)P & \text{(Struct Res Res)} \\
(\forall n)(P \mid Q) & \equiv P \mid (\forall n)Q & \text{if } n \notin \text{fn}(P) & \text{(Struct Res Par)} \\
(\forall n)(m[P]) & \equiv m[(\forall n)P] & \text{if } n \neq m & \text{(Struct Res Amb)} \\
P \mid Q & \equiv Q \mid P & \text{(Struct Par Comm)} \\
(P \mid Q) \mid R & \equiv P \mid (Q \mid R) & \text{(Struct Par Assoc)} \\
P \mid 0 & \equiv P & \text{(Struct Par Zero)} \\
!(P \mid Q) & \equiv !P \mid !Q & \text{(Struct Repl Par)} \\
!0 & \equiv 0 & \text{(Struct Repl Zero)} \\
!P & \equiv P \mid !P & \text{(Struct Repl Copy)} \\
!P & \equiv !!P & \text{(Struct Repl Repl)}
\end{align*}
\]

- These axioms (particularly the ones for \(!\)) are sound and complete with respect to equality of spatial trees; edge-labeled finite-depth unordered trees, with infinite-branching but finitely many distinct labels under each node.
### Satisfaction: Basic Tree Formulas

<table>
<thead>
<tr>
<th>Formula</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \models 0$</td>
<td>$\triangleq P \equiv 0$</td>
</tr>
<tr>
<td>$P \models n[A]$</td>
<td>$\triangleq \exists P' \in \Pi. P \equiv n[P'] \land P' \models A$</td>
</tr>
<tr>
<td>$P \models A \mid B$</td>
<td>$\triangleq \exists P', P'' \in \Pi. P \equiv P' \mid P'' \land P' \models A \land P'' \models B$</td>
</tr>
<tr>
<td>$P \models A@n$</td>
<td>$\triangleq n[P] \models A$</td>
</tr>
<tr>
<td>$P \models A\triangleright B$</td>
<td>$\triangleq \forall P' \in \Pi. P' \models A \Rightarrow P \mid P' \models B$</td>
</tr>
</tbody>
</table>

- **0**: there is no structure here now.
- **$n[A]$**: there is a location $n$ with contents satisfying $A$.
- **$A \mid B$**: there are two structures satisfying $A$ and $B$.
- **$A@n$**: when the current structure is placed in a location $n$, the resulting structure satisfies $A$.
- **$A\triangleright B$**: when the current structure is composed with one satisfying $A$, the resulting structures satisfies $B$. 
Meaning of Formulas: Satisfaction Relation

\[ P \models T \]
\[ P \models \neg A \triangleq \neg P \models A \]
\[ P \models A \lor B \triangleq P \models A \lor P \models B \]
\[ P \models 0 \triangleq P \equiv 0 \]
\[ P \models n[A] \triangleq \exists P' \in \Pi. P \equiv n[P'] \land P' \models A \]
\[ P \models A @ n \triangleq n[P] \models A \]
\[ P \models A \mid B \triangleq \exists P', P'' \in \Pi. P \equiv P' \mid P'' \land P' \models A \land P'' \models B \]
\[ P \models A \triangleright B \triangleq \forall P' \in \Pi. P' \models A \Rightarrow P \mid P' \models B \]
\[ P \models n \circ A \triangleq \exists P' \in \Pi. P \equiv (\forall n)P' \land P' \models A \]
\[ P \models A \circ n \triangleq (\forall n)P \models A \]
\[ P \models \star A \triangleq \exists P' \in \Pi. P \downarrow *P' \land P' \models A \]
\[ P \models \Diamond A \triangleq \exists P' \in \Pi. P \rightarrow *P' \land P' \models A \]
\[ P \models \forall x. A \triangleq \forall m \in \Lambda. P \models A \{x \leftarrow m\} \]

\[ P \downarrow P' \ \text{iff} \ \exists n, P''. P \equiv n[P'] \mid P''; \ \downarrow^* \ \text{is the refl-trans closure of} \ \downarrow \]
Basic Fact

- Satisfaction is invariant under structural congruence:

\[ P \models \mathcal{A}, \ P \equiv P' \Rightarrow P' \models \mathcal{A} \]

I.e.: \( \{P \in \Pi \mid P \models \mathcal{A}\} \) is closed under \( \equiv \).

- Hence, formulas describe congruence-invariant properties.
  - In particular, formulas describe properties of spatial trees.
  - N.B.: Most process logics describe bisimulation-invariant properties.

- Hence, formulas talk about trees.
From Satisfaction to (Propositional) Logic

- **Propositional validity**
  \[ \text{vld} \mathcal{A} \iff \forall P \in \Pi. P \vdash \mathcal{A} \quad \mathcal{A} \text{ (closed) is valid} \]

- **Sequents**
  \[ \mathcal{A} \vdash B \iff \forall P \in \Pi. P \vdash \mathcal{A} \Rightarrow P \vdash B \]

- **Rules**
  \[ \mathcal{A}_1 \vdash B_1; \ldots; \mathcal{A}_n \vdash B_n \vdash \mathcal{A} \vdash B \iff (n \geq 0) \]
  \[ \mathcal{A}_1 \vdash B_1 \land \ldots \land \mathcal{A}_n \vdash B_n \Rightarrow \mathcal{A} \vdash B \]
  (N.B.: all the rules shown later are validated accordingly.)

- **Conventions:**
  - \[ \vdash \] means \[ \vdash \] in both directions
  - \[ \{\{\} \] means \[ \} \} in both directions
Obtaining...

- Logical axioms and rules.
  - Rules of propositional logic (standard).
  - Rules of location and composition
    \[ \frac{\mathcal{A} \vdash C \vdash B}{\mathcal{A} \vdash C \rightarrow B} \vdash \text{adjunction} \]
  - Rules of revelation
    \[ \eta \circ \mathcal{A} \vdash B \vdash \mathcal{A} \vdash B \circ \eta \]
    \[ (\neg \mathcal{A}) \circ x \vdash \neg (\mathcal{A} \circ x) \]
    \( \circ \) is self-dual
  - Rules of \( \lozenge \) and \( \Diamond \) modalities (standard S4, plus some)
  - Rules of quantification (standard, but for name quantifiers)
- A large collection of logical consequences.
**Rules: Propositional Calculus**

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A-L)</td>
<td>$\mathcal{A} \land (\mathcal{C} \land \mathcal{D}) \vdash \mathcal{B}$  \quad (\mathcal{A} \land \mathcal{C}) \land \mathcal{D} \vdash \mathcal{B}</td>
</tr>
<tr>
<td>(A-R)</td>
<td>$\mathcal{A} \vdash (\mathcal{C} \lor \mathcal{D}) \lor \mathcal{B}$  \quad $\mathcal{A} \vdash \mathcal{C} \lor (\mathcal{D} \lor \mathcal{B})$</td>
</tr>
<tr>
<td>(X-L)</td>
<td>$\mathcal{A} \land \mathcal{C} \vdash \mathcal{B}$  \quad $\mathcal{C} \land \mathcal{A} \vdash \mathcal{B}$</td>
</tr>
<tr>
<td>(X-R)</td>
<td>$\mathcal{A} \vdash \mathcal{C} \lor \mathcal{B}$  \quad $\mathcal{A} \vdash \mathcal{B} \lor \mathcal{C}$</td>
</tr>
<tr>
<td>(C-L)</td>
<td>$\mathcal{A} \land \mathcal{A} \vdash \mathcal{B}$  \quad $\mathcal{A} \vdash \mathcal{B}$</td>
</tr>
<tr>
<td>(C-R)</td>
<td>$\mathcal{A} \vdash \mathcal{B} \lor \mathcal{B}$  \quad $\mathcal{A} \vdash \mathcal{B}$</td>
</tr>
<tr>
<td>(W-L)</td>
<td>$\mathcal{A} \vdash \mathcal{B}$  \quad $\mathcal{A} \land \mathcal{C} \vdash \mathcal{B}$</td>
</tr>
<tr>
<td>(W-R)</td>
<td>$\mathcal{A} \vdash \mathcal{B}$  \quad $\mathcal{A} \vdash \mathcal{C} \lor \mathcal{B}$</td>
</tr>
<tr>
<td>(Id)</td>
<td>$\mathcal{A} \vdash \mathcal{A}$</td>
</tr>
<tr>
<td>(Cut)</td>
<td>$\mathcal{A} \vdash \mathcal{C} \lor \mathcal{B}$; $\mathcal{A} \land \mathcal{C} \vdash \mathcal{B}'$  \quad $\mathcal{A} \land \mathcal{A}' \vdash \mathcal{B} \lor \mathcal{B}'$</td>
</tr>
<tr>
<td>(T)</td>
<td>$\mathcal{A} \land \mathcal{T} \vdash \mathcal{B}$  \quad $\mathcal{A} \vdash \mathcal{B}$</td>
</tr>
<tr>
<td>(F)</td>
<td>$\mathcal{A} \vdash \mathcal{F} \lor \mathcal{B}$  \quad $\mathcal{A} \vdash \mathcal{B}$</td>
</tr>
<tr>
<td>(∨-L)</td>
<td>$\mathcal{A} \vdash \mathcal{C} \lor \mathcal{B}$  \quad $\mathcal{A} \land \mathcal{\neg C} \vdash \mathcal{B}$</td>
</tr>
<tr>
<td>(∨-R)</td>
<td>$\mathcal{A} \land \mathcal{C} \vdash \mathcal{B}$  \quad $\mathcal{A} \vdash \mathcal{\neg C} \lor \mathcal{B}$</td>
</tr>
</tbody>
</table>
Rules: Composition

(1 0) \{ A | 0 \vdash A \} 0 is nothing

(1 \neg 0) \{ A | \neg 0 \vdash \neg 0 \} if a part is non-0, so is the whole

(A |) \{ A | (B | C) \vdash (A | B) | C \} \| associativity

(X |) \{ A | B \vdash B | A \} \| commutativity

(1 \vdash) A \vdash B; A'' \vdash B'' \{ A | A'' \vdash B' | B'' \} \| congruence

(1 \lor) \{ (A \lor B) | C \vdash A | C \lor B | C \} \| \lor \lor distribution

(1 \land) \{ A | A'' \vdash A | B'' \lor B | A'' \lor \neg B' | \neg B'' \} decomposition

(1 \Rightarrow) A | C \vdash B \{ \} A \vdash C \Rightarrow B \| \Rightarrow adjunction

(\Rightarrow F \neg) \{ A^F \vdash A^\neg \} if A is unsatisfiable then A is false

(\neg \Rightarrow F) \{ A^\neg \vdash A^{FF} \} if A is satisfiable then A^F is unsatisfiable

where \ A^\neg \triangleq \neg A \ and \ A^F \triangleq A \Rightarrow F
The Composition Adjunct

(1≥) \( \mathcal{A} \vdash C \dashv \mathcal{B} \) \( \iff \) \( \mathcal{A} \vdash C \triangleright \mathcal{B} \)

“Assume that every process that has a partition into pieces that satisfy \( \mathcal{A} \) and \( C \), also satisfies \( \mathcal{B} \). Then, every process that satisfies \( \mathcal{A} \), together with any process that satisfies \( C \), satisfies \( \mathcal{B} \). (And vice versa.)” (c.f. (\( \neg \circ \) \( R \))

- **Interpretations of \( \mathcal{A} \triangleright \mathcal{B} \):**
  - \( P \) provides \( \mathcal{B} \) in any context that provides \( \mathcal{A} \)
  - \( P \) ensures \( \mathcal{B} \) under any attack that ensures \( \mathcal{A} \)

That is, \( P \models \mathcal{A} \triangleright \mathcal{B} \) is a context-system spec (a concurrent version of a pre-post spec).

Moreover \( \mathcal{A} \triangleright \mathcal{B} \) is, in a precise sense, linear implication: the context that satisfies \( \mathcal{A} \) is used exactly once in the system that satisfies \( \mathcal{B} \).
Some Derived Rules

\[ (A \rightarrow B) \mid A \vdash B \]

“If \( P \) provides \( B \) in any context that provides \( A \), and \( Q \) provides \( A \), then \( P \) and \( Q \) together provide \( B \).”

- Proof:
  \[ A \vdash A \rightarrow B \]
  \[ (A \rightarrow B) \mid A \vdash B \]
  by (Id), (\( \vdash \))

\[ D \vdash A; \ B \vdash C \]

\[ D \mid (A \rightarrow B) \vdash C \]

(c.f. (\( \neg \rightarrow L \)))

“If anything that satisfies \( D \) satisfies \( A \), and anything that satisfies \( B \) satisfies \( C \), then: anything that has a partition into a piece satisfying \( D \) (and hence \( A \)), and another piece satisfying \( B \) in a context that satisfies \( A \), it satisfies \( B \) and hence \( C \).”

- Proof:
  \[ D \vdash A; \ A \rightarrow B \vdash A \rightarrow B \]
  \[ D \mid A \rightarrow B \vdash A \mid A \rightarrow B \]
  assumption, (Id), (\( \vdash \))

\[ A \mid A \rightarrow B \vdash B \]

above

\[ B \vdash C \]

assumption
More Derived Rules

\{ A \vdash T \mid A \}  
\text{you can always add more pieces (if they are 0)}

\{ F \mid A \vdash F \}  
\text{if a piece is absurd, so is the whole}

\{ 0 \vdash \neg(\neg 0 \mid \neg 0) \}  
0 \text{ is single-threaded}

\{ A \mid B \land 0 \vdash A \}  
\text{you can split 0 (but you get 0). Proof uses (11)}

\{ A \vdash A; B \vdash B' \}  \{ A \vdash B \vdash A \vdash B' \}  
\text{is contravariant on the left}

\{ A \vdash B \mid B \vdash C \vdash A \vdash C \}  
\text{is transitive}

\{ (A \mid B) \vdash C \vdash A \vdash (B \vdash C) \}  
\text{curry/uncurry}

\{ A \vdash (B \vdash C) \vdash B \vdash (A \vdash C) \}  
contexts commute

\{ T \vdash T \triangleright T \}  
\text{truth can withstand any attack}

\{ T \vdash F \triangleright A \}  
\text{anything goes if you can find an absurd partner}

\{ T \triangleright A \vdash A \}  
\text{if } A \text{ resists any attack, then it holds}
### Rules: Location

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>((n[] \rightarrow 0))</td>
<td>(n[A] \vdash \neg 0) locations exist</td>
</tr>
<tr>
<td>((n[] \rightarrow \bot))</td>
<td>(n[A] \vdash \neg(\neg 0 \lor \neg 0)) are not decomposable</td>
</tr>
<tr>
<td>((n[] \vdash))</td>
<td>(A \vdash B \rightarrow n[A] \vdash n[B]) (n[]) congruence</td>
</tr>
<tr>
<td>((n[] \land))</td>
<td>(n[A] \land n[C] \vdash n[A \land C]) (n[])-(\land) distribution</td>
</tr>
<tr>
<td>((n[] \lor))</td>
<td>(n[C \lor B] \vdash n[C] \lor n[B]) (n[])-(\lor) distribution</td>
</tr>
<tr>
<td>((n[] \rightarrow))</td>
<td>(n[A] \vdash B \rightarrow A \vdash B \rightarrow n) (n[])-(\rightarrow) adjunction</td>
</tr>
<tr>
<td>((\neg \rightarrow))</td>
<td>(A \rightarrow n \vdash \neg(\neg A)@n) @ is self-dual</td>
</tr>
</tbody>
</table>
Some Derived Rules

\[
A \vdash B \quad \{ \quad A \at n \vdash B \at n \quad \}
\]

\text{@ congruence}

\[
\begin{align*}
\{ & \quad n[A \at n] \vdash A \\
\{ & \quad A \vdash n[A] \at n \\
\{ & \quad n[\neg A] \vdash \neg n[A] \\
\{ & \quad \neg n[A] \vdash \neg n[T] \lor n[\neg A] \\
\end{align*}
\]
### Rules: Time and Space Modalities

<table>
<thead>
<tr>
<th>(◊)</th>
<th>◊A \models \neg \square \neg A</th>
</tr>
</thead>
<tbody>
<tr>
<td>(□ K)</td>
<td>□(A \rightarrow B) \models □A \rightarrow □B</td>
</tr>
<tr>
<td>(□ T)</td>
<td>□A \models A</td>
</tr>
<tr>
<td>(□ 4)</td>
<td>□A \models \square \square A</td>
</tr>
<tr>
<td>(□ T)</td>
<td>T \models □T</td>
</tr>
<tr>
<td>(□ t)</td>
<td>A \models B \models □A \rightarrow □B</td>
</tr>
<tr>
<td>(◊ n[])</td>
<td>n[◊A] \models ◊n[A]</td>
</tr>
<tr>
<td>(◊ 1)</td>
<td>◊A \models ◊B \models ◊(A \models B)</td>
</tr>
<tr>
<td>(◊ ◊)</td>
<td>◊ ◊A \models ◊ ◊A</td>
</tr>
</tbody>
</table>

S4, but not S5:  
\[ \neg vld \Diamond A \models \Box \Diamond A \]  
\[ \neg vld \Diamond A \models \Box \Diamond A \]

(◊ ◊): if somewhere sometime A, then sometime somewhere A
Equality

• Name equality can be defined within the logic:

\[ \eta = \mu \iff \eta[T]@\mu \]

Since (for any substitution applied to \( \eta, \mu \)):

\[ P \models \eta[T]@\mu \]

iff \( \mu[P] \models \eta[T] \)

iff \( \eta = \mu \land P \models T \)

iff \( \eta = \mu \)

• Example: “Any two ambients here have different names”:

\[ \forall x. \forall y. x[T] \parallel y[T] \parallel T \Rightarrow \neg x=y \]
Ex: Immovable Object vs. Irresistible Force

\[ Im \equiv T \triangleright \Box (obj[] \mid T) \]
\[ Ir \equiv T \triangleright \Diamond \neg (obj[] \mid T) \]

\[ Im \mid Ir \vdash (T \triangleright \Box (obj[] \mid T)) \mid T \]
\[ \vdash \Box (obj[] \mid T) \]
\[ \vdash \Diamond \Box (obj[] \mid T) \]

\[ Im \mid Ir \vdash T \mid (T \triangleright \Diamond \neg (obj[] \mid T)) \]
\[ \vdash \Diamond \neg (obj[] \mid T) \]
\[ \vdash \neg \Diamond \Box (obj[] \mid T) \]

Hence: \[ Im \mid Ir \vdash F \]
Restriction

- \((\forall n)P\)
  - “The name \(n\) is known only inside \(P\).”
  - “Create a new name \(n\) and use it in \(P\).”
  - It *extrudes* (floats) because it represents knowledge, not behavior:

  \[
  \begin{align*}
  (\forall n)P & \equiv (\forall m)(P[n \leftarrow m]) \\
  (\forall n)0 & \equiv 0 \\
  (\forall n)(\forall m)P & \equiv (\forall m)(\forall n)P \\
  (\forall n)(P \mid Q) & \equiv (\forall n)P \mid Q \quad \text{if } n \notin \text{fn}(Q) \\
  \text{a.k.a. } (\forall n)(P \mid (\forall n)Q') & \equiv (\forall n)P \mid (\forall n)Q' \\
  (\forall n)(m[P]) & \equiv m[(\forall n)P] \quad \text{if } n \neq m
  \end{align*}
  \]

- Used initially to represent private channels.
- Later, to represent private names of any kind:
  - Channels, Locations, Nonces, Cryptokeys, …
Revelation

\[ P \models n \mathop{\mathcal{R}} \mathcal{A} \triangleq \exists P' \in \Pi. P \equiv (\forall n)P' \land P' \models \mathcal{A} \]

- \( n \mathop{\mathcal{R}} \mathcal{A} \) is read, informally:
  - *Reveal* a private name as \( n \) and check that the revealed process satisfies \( \mathcal{A} \).
  - Pull out (by extrusion) a \((\forall n)\) binder, and check that the process stripped of the binder satisfies \( \mathcal{A} \).

- Examples:
  - \( n \mathop{\mathcal{R}} n[0] \): reveal a restricted name (say, \( p \)) as \( n \) and check the presence of an empty \( n \) location in the revealed process.

\[
(\forall p)p[0] \models n \mathop{\mathcal{R}} n[0] \\
\text{because } (\forall p)p[0] \equiv (\forall n)n[0] \text{ and } n[0] \models n[0]
\]
Derived Formulas: Revelation

\[ \forall n \iff \neg n \circ T \]

\[ P \vdash - \iff \neg \exists P' \in \Pi. P \equiv (\forall n)P' \]

\[ \text{iff } n \in \text{fn}(P) \]

\[ \text{closed } \iff \neg \exists x. \circ x \]

\[ P \vdash - \iff \neg \exists n \in \Lambda. n \in \text{fn}(P) \]

\[ \text{separate } \iff \neg \exists x. \circ x | \circ x \]

\[ P \vdash - \iff \neg \exists n \in \Lambda, P' \in \Pi, P'' \in \Pi. \]

\[ P \equiv P' \upharpoonright P'' \land n \in \text{fn}(P') \land n \in \text{fn}(P'') \]

- **Examples:**
  - \[ n[] \vdash \forall n \]
  - \[ (\forall p)p[] \vdash \text{closed} \]
  - \[ n[] \upharpoonright m[] \vdash \text{separate} \]
Revelation Rules

• Some mirror properties of restriction:
  \[
  \begin{align*}
  & x \Rightarrow x \Rightarrow A \vdash x \Rightarrow A \\
  & x \Rightarrow y \Rightarrow A \vdash y \Rightarrow x \Rightarrow A \\
  & x \Rightarrow (A \mid x \Rightarrow B) \vdash x \Rightarrow A \mid x \Rightarrow B \\
  \end{align*}
  \]
  (scope extrusion)

• Some behave well with logical operators:
  \[
  \begin{align*}
  & x \Rightarrow (A \lor B) \vdash x \Rightarrow A \lor x \Rightarrow A \\
  & A \vdash B \Rightarrow x \Rightarrow A \vdash x \Rightarrow B \\
  \end{align*}
  \]

• Some deal with the adjunction:
  \[
  \begin{align*}
  & \eta \Rightarrow A \vdash B \Rightarrow \{ \Rightarrow A \vdash B \Rightarrow \eta \} \\
  & (\neg A) \Rightarrow x \vdash \neg (A \Rightarrow x) \\
  & (A \mid B) \Rightarrow x \vdash A \Rightarrow x \mid B \Rightarrow x \\
  & x \Rightarrow ((A \mid B) \Rightarrow x) \vdash x \Rightarrow (A \Rightarrow x) \mid x \Rightarrow (B \Rightarrow x) \\
  \end{align*}
  \]
## Rules: Revelation

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(®)</td>
<td>( x \circ x \circ A \vdash x \circ A )</td>
</tr>
<tr>
<td>(® ®)</td>
<td>( x \circ y \circ A \vdash y \circ x \circ A )</td>
</tr>
<tr>
<td>(® ∨)</td>
<td>( x \circ (A \lor B) \vdash x \circ A \lor x \circ A )</td>
</tr>
<tr>
<td>(® ⊥)</td>
<td>( A \vdash B \vdash x \circ A \vdash x \circ B )</td>
</tr>
<tr>
<td>(® ⊙)</td>
<td>( \eta \circ A \vdash B \vdash {} A \vdash B \circ \eta )</td>
</tr>
<tr>
<td>(⊙ ¬)</td>
<td>( (\neg A) \circ x \vdash \neg (A \circ x) )</td>
</tr>
<tr>
<td>(⊙ ⊸ F)</td>
<td>( A \circ F \circ x \vdash A \circ F )</td>
</tr>
</tbody>
</table>
\((\textcircled{0})\) \{ x\textcircled{0} \vdash 0 \} \quad \text{®/☐-0 rules}

\((\text{☐ 0})\) \{ 0\text{☐}x \vdash 0 \}

\((\text{®} \text{☐} 1)\) \{ x\textcircled{1}(A \text{☐} x\textcircled{B}) \vdash x\textcircled{A} \text{☐} x\textcircled{B} \} \quad \text{®/☐-1 rules}

\((\text{☐} \text{☐} 1)\) \{ (A \text{☐} B)\text{☐}x \vdash A\text{☐}x \text{☐} B\text{☐}x \}

\((\text{®} \text{☐} \text{☐} 1)\) \{ x\textcircled{3}((A \text{☐} B)\text{☐}x) \vdash x\textcircled{A}(A\text{☐}x) \text{☐} x\textcircled{B}(B\text{☐}x) \}

\((\text{®} n[[]])\) \{ x\textcircled{n}[A] \vdash y[x\textcircled{n}A] \} \quad (x \neq y) \quad \text{®/☐-n[[-] rules}

\((\text{☐} n[[]])\) \{ y[A]\text{☐}x \vdash y[A\text{☐}x] \} \quad (x \neq y)

\((\text{☐} n[[]])\) \{ x[A]\text{☐}x \vdash \text{F} \}
Fresh-Name Quantifier

\[ P \vdash \forall x. \mathcal{A} \iff \exists m \in \Lambda. m \notin \text{fn}(P, \mathcal{A}) \land P \vdash \mathcal{A}\{x \leftarrow m\} \]

- C.f.: \( P \vdash \exists x. \mathcal{A} \iff \exists m \in \Lambda. P \vdash \mathcal{A}\{x \leftarrow m\} \)

- Actually definable (metatheoretically, as an abbreviation):
  \[
  \forall x. \mathcal{A} \iff \exists x. x#(\text{fn}(\mathcal{A})\{-x\}) \land x \in \text{T} \land \mathcal{A}
  \]
  Provided we add the axiom schema:
  \[
  (GP) \quad \exists x. x \notin N \land x \in \text{T} \land \mathcal{A} \vdash \forall x. (x \notin N \land x \in \text{T}) \Rightarrow \mathcal{A}
  \]
  where \( N \supseteq \text{fn}(\mathcal{A})\{-x\} \) and \( x \notin N \)

- Fundamental “freshness” property (Gabbay-Pitts):
  \[
  \forall x. \mathcal{A} \iff \exists m \in \Lambda. m \notin \text{fn}(P, \mathcal{A}) \land P \vdash \mathcal{A}\{x \leftarrow m\}
  \]
  \[
  \iff \forall m \in \Lambda. m \notin \text{fn}(P, \mathcal{A}) \Rightarrow P \vdash \mathcal{A}\{x \leftarrow m\}
  \]

  because any fresh name is as good as any other.
Very nice logical properties:

- $\forall x. A \vdash \forall x. A \vdash \exists x. A$
- $\neg \forall x. A \vdash \forall x. \neg A$

$\forall x. (A \lor B) \vdash (\forall x. A) \lor (\forall x. B)$  (hint: (GP) $\exists$ for $\Rightarrow$, $\forall$ for $\Leftarrow$)

- $\Diamond \forall x. A \vdash \forall x. \Diamond A$
Hidden-Name Quantifier

\[ \text{Hx} \mathcal{A} \triangleq \forall x. x \circ \mathcal{A} \]

\[ P \models \text{Hx} \mathcal{A} \quad \text{iff} \]
\[ \exists m \in \Lambda, P' \in \Pi. \quad m \notin \text{fn}(\mathcal{A}) \land P \equiv (\forall m)P' \land P' \models \mathcal{A}\{x \leftarrow m\} \]

- Example: \( \text{Hx} \cdot x[] = \forall x. x \circ x[] \)
  - “for hidden \( x \), we find a void location called \( x \)” = “for fresh \( x \), we reveal a hidden name as \( x \), then we find a void location \( x \)”
  - \( (\forall n)n[] \models \text{Hx} \cdot x[] \) because \( (\forall n)n[] \models \forall x. x \circ x[] \)
    because \( (\forall n)n[] \models n \circ n[] \) (where \( n \notin \text{fn}((\forall n)n[]) \)).

- Counterexamples:
  - \( (\forall m)m[] \not\models \text{Hx} \cdot n[] \) (N.B.: this holds for \( \text{Hx} \mathcal{A} \triangleq \exists x. x \circ \mathcal{A} !) \)
  - \( (\forall n)n[] \mid (\forall n)n[] \not\models \text{Hx} \cdot (x[] \mid x[]) \)
  - \( (\forall n)(n[] \mid n[]) \not\models \text{Hx} \cdot x[] \mid \text{Hx} \cdot x[] \)
Forget $n\text{®}_A$ and $\forall x.A$, why not just use $Hx.A$?

- Consider:
  
  $\forall x.(\forall x.A \mid x\text{®}B) 
  \vdash \forall x.(x\text{®}A \mid x\text{®}B)$
  
  $\vdash (\forall x.x\text{®}A) \mid (\forall x.x\text{®}B)$

- That is:
  
  $Hx.(A \mid x\text{®}B) \vdash Hx.A \mid Hx.B$

- Hence, the scope extrusion rule for $H$ still uses $\text{®}$.

- No matter what one chooses as primitives, we have explored interesting connections between these operators. ($\forall+\text{®}$ and $H+\text{©}$ are almost interdefinable [Caires].)
Example: Key Sharing

Consider a situation where “a hidden name $x$ is shared by two locations $n$ and $m$, and is not known outside those locations”.

$$Hx.(n[\Diamond x] | m[\Diamond x])$$

- $P \models Hx.(n[\Diamond x] | m[\Diamond x])$

  $$\Leftrightarrow \exists r \in \Lambda. r \notin fn(P) \cup \{n,m\} \land \exists R', R'' \in \Pi. P \equiv (\forall r)(n[R'] | m[R'']) \land r \in fn(R') \land r \in fn(R'')$$

- E.g.: take $P = (\forall p) (n[p[]] | m[p[]])$.

- A protocol establishing a shared key should satisfy:

  $$\Diamond Hx.(n[\Diamond x] | m[\Diamond x])$$
A logic is *just a very rich type system*.  
Type systems are very “structural” (i.e., the structure of types reflects closely the structure of values). Our logic is extremely structural (intensional) for a logic. It is in fact almost as structural as a type system.

Type systems for process calculi often have a parallel composition operation on types. I.e., they are “spatial” in our sense.

Therefore, our work may help in discerning patterns in the large and diverse collection of type systems for process calculi. These usually become particularly tangled when trying to handle restriction.

Suppose that $P : \mathcal{A}$ means that process $P$ may have “effects” $\mathcal{A}$, where an effect is any kind of information about the behavior of $P$ that one may want to track statically. Then the following kind of typing rules happen:

- Effects may be composed:
  \[
  \Gamma \vdash P : \mathcal{A}, \quad \Gamma \vdash Q : \mathcal{B} \quad \Rightarrow \quad \Gamma \vdash P | Q : \mathcal{A} \upharpoonright \mathcal{B}
  \]

- Effects may be hidden:
  \[
  \Gamma, n: \mathcal{A} \vdash P : \mathcal{B}[x \leftarrow n] \quad \Rightarrow \quad \Gamma \vdash (\nu n: \mathcal{A})P : Hx : \mathcal{A} \mathcal{B}
  \]

Applications

- **Modelchecking security+mobility assertions:**
  - If $P$ is $!$-free and $\mathcal{A}$ is $\triangleright$-free, then $P \models \mathcal{A}$ is decidable.
  - This provides a way of mechanically checking (certain) assertions about (certain) mobile processes.
  - Expressing mobility/security policies of host sites. (Conferring more flexibility than just sandboxing the agent.)
  - Just-in-time verification of code containing mobility instructions (by either modelchecking or proof-carrying code).

- **Expressing properties of type systems (beyond subject reduction).**
  - Expressing Locking
    - If $E, n:Amb^*[S] \vdash P : T$ (a typing judgment asserting that no ambient called $n$ can ever be opened in $P$), then:
      \[
P \models \Box (\Diamond an \ Leftrightarrow \Box \Diamond an)
      \]
  - Expressing Immobility
    - If $E, p:Amb^*[S], q:Amb^*[\check{S}'] \vdash P : T$ (a typing judgment asserting that no ambient called $q$ can ever move within $P$), then:
      \[
P \models \Box (\Diamond (p \text{ parents } q) \Rightarrow \Box \Diamond (p \text{ parents } q))
      \]
      where $p \text{ parents } q \triangleq p[q[T] \mid T] \mid T$
Conclusions

- The novel aspects of our logic lie in its explicit treatment of space and of the evolution of space over time (mobility).
- We can now talk also about fresh and hidden locations.
- These ideas can be applied to any process calculus that embodies a distinction between spatial and temporal operators, and a restriction operator.
- Our logical rules arise from a particular model. This approach makes the logic very concrete (and sound), but raises questions of logical completeness.

<http://www.luca.demon.co.uk> Logical Properties of Name Restriction
Exercise

• Show that \( \{ (A \mid B) \land 0 \vdash A \} \) is valid (by applying the definition of sequent and of satisfaction). The proof is short. In the process, you will discover you need a little ambient calculus lemma about \( P \mid Q \equiv 0 \); you do not need to prove it but you need to identify it.

• (Hard/Optional)
Find a formal derivation of \( \{ (A \mid B) \land 0 \vdash A \} \) from the axioms in the slides.
(My) proof uses the decomposition axiom, \(( | || )\).
END
Semantics

- **Version 1 [Cardelli-Gordon]**
  - For the restriction-free ambient calculus.
  - Formulas denote sets of processes that are closed under structural congruence.

- **Version 2 [Cardelli-Gordon]**
  - For the ambient calculus with restriction.
  - Formulas denote sets of processes that are closed under structural congruence. Freshness handled “metatheoretically”.

- **Version 3 [Caires-Cardelli]**
  - To handle both restriction and recursive formulas, and to handle freshness “properly”. (For the π-calculus, for simplicity.)
  - Formulas denote sets of processes that are closed under congruence and that have *finite support* (are closed under transpositions outside of a finite set \( N \) of names).
A Good Property

- A property not shared by other candidate definitions, such as $\exists x.x\mathcal{R}\mathcal{A}$ and $\forall x.x\mathcal{R}\mathcal{A}$. This is even derivable within the logic:

$$
Hx.(\mathcal{A}\{n\leftarrow x\}) \land n\mathcal{T} \vdash n\mathcal{R}\mathcal{A} \quad \text{where} \ x \notin \text{fv}(\mathcal{A})
$$

- It implies:

$$
\begin{align*}
P \models \mathcal{A} & \implies (\forall n)P \models Hx.(\mathcal{A}\{n\leftarrow x\}) \\
P \models Hx.(\mathcal{A}\{n\leftarrow x\}) \land n \notin \text{fn}(P) & \implies P \models n\mathcal{R}\mathcal{A} \\
P \models n\mathcal{R}\mathcal{A} & \implies P \models Hx.(\mathcal{A}\{n\leftarrow x\})
\end{align*}
$$
A Surprising Property

\[ Hx.\mathcal{A} \not\models \mathcal{A} \quad \text{for} \ x \notin \text{fv}(\mathcal{A}) \]

- **Ex.:** \( Hx.(\neg 0 \mid \neg 0) \not\models \neg 0 \mid \neg 0 \)

  If for a hidden \( x \) the inner system can be decomposed into two non-void parts, it does not mean that the whole system can be decomposed, because the two parts may be entangled by restriction:

  \[
  (\forall n)(n[] \mid n[]) \not\models \forall x.x\mathbb{R}^\circ(\neg 0 \mid \neg 0) \quad \text{but:} \quad (\forall n)(n[] \mid n[]) \not\models \neg 0 \mid \neg 0.
  \]

- **This is \( \mathbb{R} \)'s fault, not \( \forall \)'s:** with the same counterexample we can show \( n\mathbb{R}(\neg 0 \mid \neg 0) \not\models \neg 0 \mid \neg 0. \)

- **However,** \( Hx.0 \vdash 0. \)

- **Moreover,** \( \mathcal{A} \vdash Hx.\mathcal{A} \) for \( x \notin \text{fv}(\mathcal{A}). \)
Satisfaction for Hidden-Name Quantification

- It makes sense also to define a *hidden name quantifier* $\text{H}x.\mathcal{A}$:
  - $n\mathcal{A}$: reveal a hidden name *if possible* as a given $n$, and assert $\mathcal{A}\{n\}$.
  - $\text{H}x.\mathcal{A}$: reveal a hidden name as *any fresh* name $x$ and assert $\mathcal{A}\{x\}$.

\[
\begin{align*}
&n \\
n \triangledown P \models \text{H}x.\mathcal{A} \\
&\text{if} \\
P \models \mathcal{A}\{x\leftarrow n\}
\end{align*}
\]

with $n \notin \text{fn}(\mathcal{A})$

- Design decision: how to define $\text{H}x.\mathcal{A}$, keeping in mind that “freshness” may spill into the logic?
  - *The Obvious Thing*: extend the syntax with $\text{H}x.\mathcal{A}$ and define it directly.
  - *Luis Caires*: Extend the syntax with $\text{H}x.\mathcal{A}$ and add signatures to keep track of free names, to enforce the side condition $n \notin \text{fn}(\mathcal{A})$: $\Sigma \bullet P \models \Sigma \bullet \mathcal{A}$.
  - *Us*: Retain $n\mathcal{A}$ and mix it with a logical notions of freshness $\forall x.\mathcal{A}$ (one extra axiom schema, no new syntax). We eventually define: $\text{H}x.\mathcal{A} \triangleq \forall x.\mathcal{A}$. 
The Decomposition Operator

- Consider the De Morgan dual of $\vdash$:

$$A || B \equiv \neg(\neg A \land \neg B)$$  

$P \vdash - \iff \forall P', P'' \in \Pi. P \equiv P' \land P'' \Rightarrow P' \vdash A \lor P'' \vdash B$

$$A^\forall \equiv A || F$$  

$P \vdash - \iff \forall P', P'' \in \Pi. P \equiv P' \land P'' \Rightarrow P' \vdash A$

$$A^\exists \equiv A || T$$  

$P \vdash - \iff \exists P', P'' \in \Pi. P \equiv P' \land P'' \land P' \vdash A$

$A || B$ for every partition, one piece satisfies $A$

or the other piece satisfies $B$

$A^\forall \iff \neg((\neg A)^\exists)$ every component satisfies $A$

$A^\exists \iff \neg((\neg A)^\forall)$ some component satisfies $A$

Examples:

$$(p[T] \Rightarrow p[q[T]^3])^\forall$$ every $p$ has a $q$ child

$$(p[T] \Rightarrow p[q[T] \land (\neg q[T])^\forall])^\forall$$ every $p$ has a unique $q$ child
The Decomposition Axiom

\[(1 \equiv) \quad (A \parallel A^\prime) \vdash (A^\prime \parallel B^\prime) \lor (B \parallel A^\prime) \lor (\neg B^\prime \parallel \neg B^\prime)\]

- Alternative formulations and special cases:

  \[(A^\prime \parallel A^\prime) \land (B^\prime \parallel B^\prime) \vdash (A^\prime \parallel B^\prime) \lor (B^\prime \parallel A^\prime)\]
  
  “If \( P \) has a partition into pieces that satisfy \( A^\prime \) and \( A^\prime \), and every partition has one piece that satisfies \( B^\prime \) or the other that satisfies \( B^\prime \), then either \( P \) has a partition into pieces that satisfy \( A^\prime \) and \( B^\prime \), or it has a partition into pieces that satisfy \( B^\prime \) and \( A^\prime \).”

  \[\neg(A \parallel B) \vdash (A \parallel T) \Rightarrow (T \parallel \neg B)\]
  
  “If \( P \) has no partition into pieces that satisfy \( A \) and \( B \), but \( P \) has a piece that satisfies \( A \), then \( P \) has a piece that does not satisfy \( B \).”

  \[\neg(T \parallel B) \vdash T \parallel \neg B\]

  \[\neg(A \parallel B) \vdash (\neg A \parallel T) \lor (T \parallel \neg B)\]
Logical Adjunctions

• This is a logic with multiple logical adjunctions (4 of them!):

  \( \land / \Rightarrow \) (classical)

  \( \mathcal{A} \land C \vdash B \iff \mathcal{A} \vdash C \Rightarrow B \)

• \( | / \triangleright \) (linear, \( \otimes / \leftarrow \))

  \( \mathcal{A} \vdash C \vdash B \iff \mathcal{A} \vdash C \triangleright B \)

• \( n[-] / -@n \)

  \( n[\mathcal{A}] \vdash B \iff \mathcal{A} \vdash B@n \)

• \( n\otimes / -\otimes n \)

  \( n\otimes \mathcal{A} \vdash B \iff \mathcal{A} \vdash B\otimes n \)

• Which one should be taken as the logical adjunction for sequents? (I.e., what should “,” mean in a sequent?)

• We do not choose, and take sequents of the form \( \mathcal{A} \vdash B \).
“Neutral” Sequents

• Our logic is formulated with single-premise, single-conclusion sequents. We don’t pre-judge “,”.

  • By taking $\land$ on the left and $\lor$ on the right of $\vdash$ as structural operators, we can derive all the standard rules of sequent and natural deduction systems with multiple premises/conclusions.

  • By taking $| -$ on the left of $\vdash$ as a structural operator, we can derive all the rules of intuitionistic linear logic (by appropriate mappings of the ILL connectives).

  • By taking nestings of $\land$ and $|$ on the left of $\vdash$ as structural “bunches”, we obtain a bunched logic, with its two associated implications, $\Rightarrow$ and $\Rightarrow$.

• This is convenient. We do not know much, however, about the meta-theory of this presentation style.
Ambient Calculus: Example

**location a**

\[ a[msg[\langle M \rangle \mid \text{out } a. \text{ in } b. \ P]] \]

**send M from a to b**

**location b**

\[ \mid b[\text{open msg. } (n). \ P] \]

**receive n; do P**

The packet `msg` moves from `a` to `b`, mediated by the capabilities `out a` (to exit `a`), `in b` (to enter `b`), and `open msg` (to open the `msg` envelope).

(\text{exit}) \rightarrow \quad a[] \quad \mid \quad msg[\langle M \rangle \mid \text{in } b. \ P] \quad \mid \quad b[\text{open msg. } (n). \ P] \\
(\text{enter}) \rightarrow \quad a[] \quad \mid \quad b[\text{msg}[\langle M \rangle] \mid \text{open msg. } (n). \ P] \\
(\text{open}) \rightarrow \quad a[] \quad \mid \quad b[\langle M \rangle \mid (n). \ P] \\
(\text{read}) \rightarrow \quad a[] \quad \mid \quad b[P \{n \leftarrow M\}]

2003-03-17 16:49
Talk 63
Connections with Intuitionistic Linear Logic

- Weakening and contraction are not valid rules: principle of conservation of space.

- Semantic connection: sets of processes closed under \( \equiv \) and ordered by inclusion form a quantale (a model of ILL).

- Multiplicative intuitionistic linear logic (MILL) can be faithfully embedded in our logic:

\[
\begin{align*}
1_{\text{MILL}} & \trianglelefteq 0 \\
\mathcal{A} \otimes_{\text{MILL}} \mathcal{B} & \trianglelefteq \mathcal{A} \triangleright \mathcal{B} \\
\mathcal{A} \rightarrow_{\text{MILL}} \mathcal{B} & \trianglelefteq \mathcal{A} \triangleright \mathcal{B}
\end{align*}
\]

MILL rules and our rules are interderivable (“our rules” means the rules involving only 0, \( \triangleright \), plus a derivable cut rule for \( \triangleright \)).
Full intuitionistic linear logic (ILL) can be embedded:

\[
\begin{align*}
1_{\text{ILL}} & \triangleq 0 \\
\bot_{\text{ILL}} & \triangleq F \\
\top_{\text{ILL}} & \triangleq T \\
0_{\text{ILL}} & \triangleq F
\end{align*}
\]

\[
\begin{align*}
A \oplus B & \triangleq A \vee B \\
A \& B & \triangleq A \wedge B \\
A \otimes B & \triangleq A \mid B \\
!A \triangleq 0 \wedge (0 \Rightarrow A)^{-F}
\end{align*}
\]

The rules of ILL can be logically derived from these definitions. (E.g.: the proof of \( !A \vdash !A \otimes !A \) uses the decomposition axiom.)

So, \( A_1, \ldots, A_n \vdash_{\text{ILL}} B \) implies \( A_1 \mid \ldots \mid A_n \vdash B \).

Some discrepancies: \( \bot_{\text{ILL}} = 0_{\text{ILL}} \); the additives distribute; \( !A \) is not “replication”; \( !A \rightarrow B \) is not so interesting; \( A_{\bot}/A^0 \) is unusually interesting.
Connection with Relevant Logic

(Noted after the fact [O’Hearn, Pym].) The definition of the satisfaction relation is very similar to Urquhart’s semantics of relevant logic. In particular \( \mathcal{A} \mid \mathcal{B} \) is defined just like intensional conjunction, and \( \mathcal{A} \rightarrow \mathcal{B} \) is defined just like relevant implication in that semantics.

Except:

- We do not have contraction. This does not make sense in process calculi, because \( P \mid P \neq P \). Urquhart semantics without contraction does not seem to have been studied.

- We use an equivalence \( \equiv \), instead of a Kripke-style partial order \( \varnothing \) as in Urquhart’s general case. (We may have a need for a partial order in more sophisticated versions of our logic.)
Connections with Bunched Logic

- Peter O’Hearn and David Pym study \textit{bunched logics}, where sequents have two structural combinators, instead of the standard single “,” combinator (usually meaning $\land$ or $\otimes$ on the left) found in most presentations of logic. Thus, sequents are \textit{bunches} of formulas, instead of lists of formulas. Correspondingly, there are two implications that arise as the adjuncts of the two structural combinators.

- The situation is very similar to our combinators $|$ and $\land$, which can combine to irreducible bunches of formulas in sequents, and to our two implications $\Rightarrow$ and $\triangleright$. However, we have a classical and a linear implication, while bunched logics have so far had an intuitionistic and a linear implication.
Semantic Connections with the Linear Logic

• A (commutative) quantale $Q$ is a structure

\[ \langle S \in \text{Set}, \leq \in S^2 \rightarrow \text{Bool}, \vee \in \mathcal{P}(S) \rightarrow S, \otimes \in S^2 \rightarrow S, 1 \in S \rangle \] such that:

- $\leq, \vee$ is a complete join semilattice
- $\otimes, 1$ is a commutative monoid

\[ p \otimes \vee Q = \vee \{ p \otimes q \mid q \in Q \} \]

• They are complete models of Intuitionistic Linear Logic (ILL):

\[
\begin{align*}
[A \oplus B] & \triangleq \vee \{ [A], [B] \} \\
[A \& B] & \triangleq \vee \{ C \mid C \leq [A] \land C \leq [B] \} \\
[A \otimes B] & \triangleq [A] \otimes [B] \\
[A \multimap B] & \triangleq \vee \{ C \mid C \otimes [A] \leq [B] \} \\
[!A] & \triangleq \nu X. [1 \& A \& X \otimes X] \quad \text{where} \quad \nu X. A\{X\} \triangleq \vee \{ C \mid C \leq A\{C\} \}
\end{align*}
\]

\[
\text{vld}_{\text{ILL}}(\mathcal{A}_1, \ldots, \mathcal{A}_n \vdash_{\text{ILL}} B)_Q \triangleq \left[ \mathcal{A}_1 \right]_Q \otimes_Q \ldots \otimes_Q \left[ \mathcal{A}_n \right]_Q \leq_Q \left[ B \right]_Q
\]
The Process Quantale

- The sets of processes closed under $\equiv$ and ordered by inclusion form a quantale (let $A^\equiv \triangleq \{ P \mid \exists Q \in A. \ P \equiv Q \}$):

  $\Theta \triangleq \langle \Phi, \subseteq, \cup, \otimes, 1 \rangle$ where, for $A, B \subseteq \Pi$:
  
  $\Phi \triangleq \{ A^\equiv \mid A \subseteq \Pi \}$
  
  $1_\Theta \triangleq \{ 0 \}^\equiv$
  
  $A \otimes_\Theta B \triangleq \{ P \mid Q \mid P \in A \land Q \in B \}^\equiv$

- ILL validity in $\Theta$:

  $\text{vld}_{\text{ILL}}(\mathcal{A}_1, \ldots, \mathcal{A}_n \vdash_{\text{ILL}} B)_\Theta$
  
  $\iff [\mathcal{A}_1] \otimes_\Theta \ldots \otimes_\Theta [\mathcal{A}_n] \subseteq [B]$
  
  $\iff [\mathcal{A}_1 \mid \ldots \mid \mathcal{A}_n] \subseteq [B]$
  
  $\iff (\Pi - [\mathcal{A}_1 \mid \ldots \mid \mathcal{A}_n]) \cup [B] = \Pi$
  
  $\iff [\mathcal{A}_1 \mid \ldots \mid \mathcal{A}_n \Rightarrow B] = \Pi$
Process Domain

• Semantic domain: Θ

\( \Pi \triangleq \text{the set of process expressions} \)

\( \forall C \subseteq \Pi. \quad C^= \triangleq \{ P \in \Pi \mid \exists P' \in C. P' \equiv P \} \)

\( \Phi \triangleq \{ C^= \mid C \subseteq \Pi \} \)

The domain Θ is both a quantale \((1, \otimes, \subseteq, \cup)\) and a boolean algebra \((\phi, \Pi, \cup, \cap, \Pi^-)\). It has additional structure induced by \(n[P]\) and \((\forall n)P\).

• Spatial operators over Θ:

\( 1 \triangleq \{ 0 \}^= \)

\( \forall C, D \in \Theta. \quad C \otimes D \triangleq \{ P \mid Q \mid P \in C \land Q \in D \}^= \)

\( \forall n \in \Lambda, \ C \in \Theta. \quad n[C] \triangleq \{ n[P] \mid P \in C \}^= \)

\( \forall n \in \Lambda, \ C \in \Theta. \quad n \cap C \triangleq \{ (\forall n)P \mid P \in C \}^= \)
Semantics of Revelation

\[ n \mathcal{C} \triangleq \{ (\forall n)P \mid P \in C \} = \]

- This means: take all processes of the form \((\forall n)P\) (not up to renaming of \(n\)), remove the ones such that \(P \notin C\), and \(\equiv\)-close the result (thus adding all the \(\alpha\)-variants).

- \(n \mathcal{C}\) is read, informally:
  - Reveal a private name as \(n\) and check that the contents are in \(C\).
  - Pull (by \(\equiv\)) a \((\forall n)\) binder at the top and check the rest is in \(C\).

- Ex.: \(n \mathcal{C}\[1]\): reveal a private name (say, \(p\)) as \(n\) and check that there is an empty \(n\) ambient in the revealed process.

\[(\forall p)p[\emptyset] \in n \mathcal{C}\[1]\]

because \((\forall p)p[\emptyset] \equiv (\forall n)n[\emptyset]\) and \(n[\emptyset] \in n[1]\)
• More examples of $n \otimes C \triangleq \{ \forall n)P \mid P \in C \}'$:  
  • $0 \in n \otimes 1$ because $0 \equiv (\forall n)0$ and $0 \in 1$
  • $m[0] \in n \otimes \Pi$ because $m[0] \equiv (\forall n)m[0]$ and $m[0] \in \Pi$
  • $n[0] \notin n \otimes \Pi$ because $n[0] \not\equiv (\forall n)...

• Therefore, $n \otimes C$ is:
  • closed under $\alpha$-variants
  • closed under $\equiv$-variants
  • not closed under changes in the set of free names
  • not closed under reduction (free names may disappear)
  • not closed under any equivalence that includes reduction
  • still ok for temporal reasoning: $\neg n \otimes A \land \diamond n \otimes A$
Semantics of the Logic

- $[T] \equiv \Pi$
- $[-A] \equiv \Pi - [A]$
- $[A \lor B] \equiv [A] \cup [B]$
- $[0] \equiv 1$
- $[n[A]] \equiv n[[A]]$
- $[A@n] \equiv \bigcup\{C \in \Theta \mid n[C] \subseteq [A]\}$
- $[A \mid B] \equiv [A] \otimes [B]$
- $[A\triangleright B] \equiv \bigcup\{C \in \Theta \mid C \otimes [A] \subseteq [B]\}$
- $[n\otimes A] \equiv n\otimes[A]$
- $[A\mathcal{O}n] \equiv \bigcup\{C \in \Theta \mid n\otimes C \subseteq [A]\}$
- $[\otimes A] \equiv \{P \in \Pi \mid \exists P' \in \Pi. P \downarrow^* P' \land P' \in [A]\}$
- $[\diamond A] \equiv \{P \in \Pi \mid \exists P' \in \Pi. P \rightarrow^* P' \land P' \in [A]\}$
- $[\forall x. A] \equiv \bigcap_{m \in A} [A\{x \leftarrow m\}]$

$P \downarrow P' \iff \exists n, P''. P \equiv n[P'] \mid P'';$ $\downarrow^*$ is the refl-trans closure of $\downarrow.$
Basic Fact

• Formulas describe only congruence-invariant properties:

∀A ∈ Φ. [A] ∈ Θ
The properties of satisfaction for each logic constructs are then derivable.

This approach to defining satisfaction is particularly good for introducing recursive formulas in the logic: it is easy to give them semantics as least and greatest fixpoints in the model, while it is not easy to define them directly via a satisfaction relation.