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 …”.
Spatial Structures

- 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]]] | ...]
```
Ambient Structures

- 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.

\[ a[c[out \ a. \ in \ b. \ P]] \mid b[0] \]
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>Process</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 \mid Q)</td>
<td>(composition)</td>
</tr>
</tbody>
</table>

**Formulas**

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

**Trees**

- \(n\)
- \(P\)
- \(P \mid Q\)

**Trees**

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

- *Mobility* is change of spatial structures over time.

\[ a[Q | c[\text{out a. in b. P}]] \quad \mid \quad b[R] \]
Mobility

- **Mobility** is change of spatial structures over time.
**Mobility**

- *Mobility* is change of spatial structures over time.

```
\begin{align*}
a & | b[R | c[P]]
\end{align*}
```
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, …
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.
Modal Logics

- In a modal logic, the truth of a formula is relative to a state (called a \textit{world}).
  - Temporal logic: current time.
  - Program logic: current store contents.
  - Epistemic logic: current knowledge. Etc.

- In our case, the truth of a \textit{space-time modal formula} is relative to the \textit{here and now} of a process.
  - The formula $n[0]$ is read:
    \begin{quote}
    there is here and now an empty location called $n$
    \end{quote}
  - The operator $n[\overline{A}]$ 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**

\[ \mathcal{A} \in \Phi ::= \text{Formulas} \quad (\eta \text{ is a name } n \text{ or a variable } x) \]

- **T** \quad true
- **\neg \mathcal{A}** \quad negation
- **\mathcal{A} \lor \mathcal{A}'** \quad disjunction
- **0** \quad void
- **\eta[\mathcal{A}]** \quad location
- **\mathcal{A} @ \eta** \quad location adjunct
- **\mathcal{A} \mid \mathcal{A}'** \quad composition
- **\mathcal{A} \triangleright \mathcal{A}'** \quad composition adjunct
- **\eta \triangleleft \mathcal{A}** \quad revelation
- **\mathcal{A} \triangleleft \eta** \quad revelation adjunct
- **\spadesuit \mathcal{A}** \quad somewhere modality
- **\diamondsuit \mathcal{A}** \quad sometime modality
- **\forall x. \mathcal{A}** \quad universal quantification over names
**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

• $an \ n \triangleq n[T] \mid T$
  
  there is now an $n$ here

• $no \ n \triangleq \neg an \ n$
  
  there is now no $n$ here

• $one \ n \triangleq n[T] \mid no \ n$
  
  there is now exactly one $n$ here

• $\forall \ A \triangleq \neg (\neg A \mid T)$
  
  everybody here satisfies $\bar{A}$

• $(n[T] \Rightarrow n[\bar{A}])\forall$
  
  every $n$ here satisfies $\bar{A}$

• $(\forall (n[T] \Rightarrow n[\bar{A}])\forall)$
  
  every $n$ everywhere satisfies $\bar{A}$
Satisfaction for Basic Trees

- $\models 0$

- $\models n[A] \quad \text{if} \quad P \models A$

- $\models A \cup B \quad \text{if} \quad P \models A \quad \text{and} \quad Q \models B$

- $\models A \cup n \quad \text{if} \quad P \models A$

- $\models A \Rightarrow B \quad \text{if for all} \quad Q \models A \quad \text{we have} \quad P \cup Q \models B$
Satisfaction for Somewhere/Sometime

\[ P \models \diamond \mathcal{A} \quad \text{if} \quad Q \models \mathcal{A} \]

\[ P \models \diamond \mathcal{A} \quad \text{if} \quad P \xrightarrow{*} Q \quad \text{and} \quad Q \models \mathcal{A} \]

- N.B.: instead of \( \diamond \mathcal{A} \) and \( \lozenge \mathcal{A} \) 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:

\[
\begin{align*}
P & \overset{m}{=} P \left\{ m \leftarrow n \right\} \\
\quad & \quad \\
(m) & \quad (n \neq m) \\
\quad & \quad \\
P & \overset{n}{=} P \\
\end{align*}
\]

Etc.

\[
\begin{align*}
P & \not\models n \otimes A \\
\quad & \quad \\
\not\models A & \quad \text{if } n \text{ is free!}
\end{align*}
\]

\[
\begin{align*}
P & \not\models A \otimes n \\
\quad & \quad \\
\not\models A & \quad \text{if } n \text{ is free!}
\end{align*}
\]
**Intended Model: Ambient Calculus**

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

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

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

\[
\begin{align*}
\text{Red In:} & \
\text{Red Out:} & \
\text{Red Open:} & \
\text{Red Comm:} & \\ 
\text{Red Res:} & \
\text{Red Amb:} & \
\text{Red Par:} & \
\text{Red \equiv:} & \\
\end{align*}
\]

\[
\begin{align*}
\text{Red In:} & \quad n[in m. P \mid Q] \mid m[R] \to m[n[P \mid Q] \mid R] \\
\text{Red Out:} & \quad m[n[out m. P \mid Q] \mid R] \to n[P \mid Q] \mid m[R] \\
\text{Red Open:} & \quad \text{open m. } P \mid m[Q] \to P \mid Q \\
\text{Red Comm:} & \quad (n).P \mid \langle M \rangle \to P\{n \leftarrow M\} \\
\text{Red Res:} & \quad P \to Q \Rightarrow (\forall n)P \to (\forall n)Q \\
\text{Red Amb:} & \quad P \to Q \Rightarrow n[P] \to n[Q] \\
\text{Red Par:} & \quad P \to Q \Rightarrow P \mid R \to Q \mid R \\
\text{Red \equiv:} & \quad P' \equiv P, P \to Q, Q \equiv Q' \Rightarrow P' \to Q'
\end{align*}
\]

\[\to^*\] is the reflexive-transitive closure of \[\to\]
Structural Congruence

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

\[
P \equiv P \quad \text{(Struct Refl)}
\]

\[
P \equiv Q \implies Q \equiv P \quad \text{(Struct Symm)}
\]

\[
P \equiv Q, Q \equiv R \implies P \equiv R \quad \text{(Struct Trans)}
\]

\[
P \equiv Q \implies (\forall n)P \equiv (\forall n)Q \quad \text{(Struct Res)}
\]

\[
P \equiv Q \implies P \parallel R \equiv Q \parallel R \quad \text{(Struct Par)}
\]

\[
P \equiv Q \implies !P \equiv !Q \quad \text{(Struct Repl)}
\]

\[
P \equiv Q \implies M[P] \equiv M[Q] \quad \text{(Struct Amb)}
\]

\[
P \equiv Q \implies M.P \equiv M.Q \quad \text{(Struct Action)}
\]

\[
P \equiv Q \implies (n).P \equiv (n).Q \quad \text{(Struct Input)}
\]

\[
\varepsilon.P \equiv P \quad \text{(Struct } \varepsilon\text{)}
\]

\[
(M.M').P \equiv M.M'.P \quad \text{(Struct .)}
\]
\[(\forall n)0 \equiv 0\]  
\[(\forall n)(\forall m)P \equiv (\forall m)(\forall n)P\]  
\[(\forall n)(P \mid Q) \equiv P \mid (\forall n)Q \quad \text{if } n \notin \text{fn}(P)\]  
\[(\forall n)(m[P]) \equiv m[(\forall n)P] \quad \text{if } n \neq m\]  
\[P \mid Q \equiv Q \mid P\]  
\[(P \mid Q) \mid R \equiv P \mid (Q \mid R)\]  
\[P \mid 0 \equiv P\]  
\[!(P \mid Q) \equiv !P \mid !Q\]  
\[!0 \equiv 0\]  
\[!P \equiv P \mid !P\]  
\[!P \equiv !!P\]

- 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

\[ P \models 0 \triangleq P \equiv 0 \]
\[ P \models n[A] \triangleq \exists P' \in \Pi. P \equiv n[P'] \land P' \models A \]
\[ P \models A \mid B \triangleq \exists P', P'' \in \Pi. P \equiv P' \mid P'' \land P' \models A \land P'' \models B \]
\[ P \models A@n \triangleq n[P] \models A \]
\[ P \models A\triangleright B \triangleq \forall P' \in \Pi. P' \models A \Rightarrow P \mid P' \models B \]

- **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

\[
P \models T
\]

\[
P \models \lnot \mathcal{A} \triangleq \lnot P \models \mathcal{A}
\]

\[
P \models \mathcal{A} \lor B \triangleq P \models \mathcal{A} \lor P \models B
\]

\[
P \models 0 \triangleq P \equiv 0
\]

\[
P \models n[\mathcal{A}] \triangleq \exists P' \in \Pi. P \equiv n[P'] \land P' \models \mathcal{A}
\]

\[
P \models \mathcal{A} @ n \triangleq n[P] \models \mathcal{A}
\]

\[
P \models \mathcal{A} \mid B \triangleq \exists P', P'' \in \Pi. P \equiv P' \mid P'' \land P' \models \mathcal{A} \land P'' \models B
\]

\[
P \models \mathcal{A} \rightarrow B \triangleq \forall P' \in \Pi. P' \models \mathcal{A} \Rightarrow P \mid P' \models B
\]

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

\[
P \models \mathcal{A} \circ n \triangleq (\forall n)P \models \mathcal{A}
\]

\[
P \models \Diamond \mathcal{A} \triangleq \exists P' \in \Pi. P \downarrow^* P' \land P' \models \mathcal{A}
\]

\[
P \models \Diamond x. \mathcal{A} \triangleq \forall m \in \Lambda. P \models \mathcal{A}\{x \leftrightarrow m\}
\]

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

• Satisfaction is invariant under structural congruence:

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

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
  \[ \begin{align*}
  \mathcal{A}_1 \vdash B_1; \ldots; \mathcal{A}_n \vdash B_n & \vdash \mathcal{A} \vdash B \iff (n \geq 0) \\
  \mathcal{A}_1 \vdash B_1 \land \ldots \land \mathcal{A}_n \vdash B_n \Rightarrow \mathcal{A} \vdash B
  \end{align*} \]

  (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
    \[ \mathcal{A} \vdash C \vdash B \quad \Rightarrow \quad \mathcal{A} \vdash C \supset B \quad \vdash \text{adjunction} \]
  - Rules of revelation
    \[ \eta \circ \mathcal{A} \vdash B \quad \Rightarrow \quad \mathcal{A} \vdash B \otimes \eta \quad \circ \cdot \text{adjunction} \]
    \[ \{ (\neg \mathcal{A}) \otimes x \vdash \neg (\mathcal{A} \otimes x) \quad \otimes \text{is self-dual} \]
  - Rules of ♠ and ◇ 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>( A \land (C \land D) \vdash B } { (A \land C) \land D \vdash B )</td>
</tr>
<tr>
<td>(A-R)</td>
<td>( A \vdash (C \lor D) \lor B } { A \vdash C \lor (D \lor B) )</td>
</tr>
<tr>
<td>(X-L)</td>
<td>( A \land C \vdash B } { C \land A \vdash B )</td>
</tr>
<tr>
<td>(X-R)</td>
<td>( A \vdash C \lor B } { A \vdash B \lor C )</td>
</tr>
<tr>
<td>(C-L)</td>
<td>( A \land A \vdash B } { A \vdash B )</td>
</tr>
<tr>
<td>(C-R)</td>
<td>( A \vdash B \lor B } { A \vdash B )</td>
</tr>
<tr>
<td>(W-L)</td>
<td>( A \vdash B } { A \land C \vdash B )</td>
</tr>
<tr>
<td>(W-R)</td>
<td>( A \vdash B } { A \vdash C \lor B )</td>
</tr>
<tr>
<td>(Id)</td>
<td>{ A \vdash A )</td>
</tr>
<tr>
<td>(Cut)</td>
<td>( A \vdash C \lor B; A \land C \vdash B' } { A \land A' \vdash B \lor B' )</td>
</tr>
<tr>
<td>(T)</td>
<td>( A \land T \vdash B } { A \vdash B )</td>
</tr>
<tr>
<td>(F)</td>
<td>( A \vdash F \lor B } { A \vdash B )</td>
</tr>
<tr>
<td>(→-L)</td>
<td>( A \vdash C \lor B } { A \land \neg C \vdash B )</td>
</tr>
<tr>
<td>(→-R)</td>
<td>( A \land C \vdash B } { A \vdash \neg C \lor B )</td>
</tr>
</tbody>
</table>
### Rules: Composition

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>((1 , 0))</td>
<td>(A \mid 0 \rightarrow A) if 0 is nothing</td>
</tr>
<tr>
<td>((1 , \neg 0))</td>
<td>(A \mid \neg 0 \rightarrow \neg 0) if a part is non-0, so is the whole</td>
</tr>
<tr>
<td>((A \mid))</td>
<td>(A \mid (B \mid C) \rightarrow (A \mid B) \mid C) (\vdash) associativity</td>
</tr>
<tr>
<td>((X \mid))</td>
<td>(A \mid B \rightarrow B \mid A) (\vdash) commutativity</td>
</tr>
<tr>
<td>((1 \neg))</td>
<td>(A' \mid B'; A'' \mid B'' \rightarrow A' \mid A'' \rightarrow B' \mid B'') (\vdash) congruence</td>
</tr>
<tr>
<td>((1 \lor))</td>
<td>((A \lor B) \mid C \rightarrow A \mid C \lor B \mid C) (\vdash) (\lor) (\lor) distribution</td>
</tr>
<tr>
<td>((1 \parallel))</td>
<td>(A \mid A'' \rightarrow A' \mid B' \lor A'' \rightarrow B' \lor \neg B' \lor \neg B'') decomposition</td>
</tr>
<tr>
<td>((1 \triangleright))</td>
<td>(A \mid C \rightarrow B) (\vdash) (\rightarrow) adjunction</td>
</tr>
<tr>
<td>(\triangleright F \neg))</td>
<td>(A^F \rightarrow A^\bot) if (A) is unsatisfiable then (A) is false</td>
</tr>
<tr>
<td>(\neg \triangleright F)</td>
<td>(A^F \rightarrow A^{FF}) if (A) is satisfiable then (A^F) is unsatisfiable</td>
</tr>
</tbody>
</table>

where \(A^\bot \iff \neg A\) and \(A^F \iff A \triangleright F\)
The Composition Adjunct

(1⇒) $\mathcal{A} \vdash C \rightarrow B \iff \mathcal{A} \vdash C \triangleright B$

“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. ($\rightarrow \neg R$))

• Interpretations of $\mathcal{A} \triangleright B$:
  • $P$ 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 \to 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: \(A \vdash \lambda A \vdash \lambda B \to A \vdash B\)  \(\vdash \lambda (A \to B) \mid A \vdash B\) by (Id), \((\vdash\lambda)\)

\(D \vdash A; \ B \vdash C \to D \mid (A \to B) \vdash C\)

“(c.f. \((\to\land)\))

“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:

\[ D \vdash A; \ \lambda A \vdash \lambda B \to A \vdash B \to D \mid \lambda (A \to B) \vdash A \mid \lambda (A \to B) \]

assumption, (Id), \((\vdash\lambda)\)

above

\[ \lambda A \lambda B \vdash B \]

assumption

\[ B \vdash C \]
More Derived Rules

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

\{ F | A \vdash F \} 
- if a piece is absurd, so is the whole

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

\{ A | B \land 0 \vdash A \} 
- you can split 0 (but you get 0). Proof uses ( || )

\begin{align*}
A \vdash A, & B \vdash B' \} \ A \triangleright B \vdash A \triangleright B' & \triangleright \text{ is contravariant on the left} \\
A \triangleright B & | B \triangleright C \vdash A \triangleright C & \triangleright \text{ is transitive} \\
( A | B ) \triangleright C & \vdash A \triangleright (B \triangleright C) & \triangleright \text{ curry/uncurry} \\
A \triangleright (B \triangleright C) & \vdash B \triangleright (A \triangleright C) & \text{ contexts commute}
\end{align*}

\begin{align*}
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}
\end{align*}
## Rules: Location

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
</table>
| $(n[] \rightarrow 0)$ | $n[\mathcal{A}] \vdash \neg 0$  
locations exist  
are not decomposable |
| $(n[] \rightarrow \bot)$ | $n[\mathcal{A}] \vdash \neg(\neg 0 \lor \neg 0)$  |
| $(n[] \vdash) \quad \mathcal{A} \vdash B$ | $\{ \} \quad n[\mathcal{A}] \vdash n[B]$  
n[] congruence  
n[]-\land distribution  
n[]-\lor distribution |
| $(n[] \land) \quad \mathcal{A} \land n[C] \vdash n[\mathcal{A} \land C]$  |
| $(n[] \lor) \quad n[C \lor B] \vdash n[C] \lor n[B]$  |
| $(n[] @) \quad n[\mathcal{A}] \vdash B$ | $\{ \} \quad \mathcal{A} \vdash B \mathcal{A}$  
n[]-@ adjunction  
@ is self-dual  
\mathcal{A} \vdash \neg((\neg \mathcal{A}) @ n)$  |

\[ \]
Some Derived Rules

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

\( n[A \vdash n] \vdash A \)

\( A \vdash n[A \vdash n] \)

\( n[\neg A] \vdash \neg n[A] \)

\( \neg n[A] \vdash \neg n[T \lor n[\neg A]] \)
### Rules: Time and Space Modalities

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(\Diamond)$</td>
<td>$\Diamond A \vdash \neg \Box \neg A$</td>
</tr>
<tr>
<td>$(\Box K)$</td>
<td>$\Box (A \rightarrow B) \vdash \Box A \rightarrow \Box B$</td>
</tr>
<tr>
<td>$(\Box T)$</td>
<td>$\Box A \vdash A$</td>
</tr>
<tr>
<td>$(\Box 4)$</td>
<td>$\Box A \vdash \Box \Box A$</td>
</tr>
<tr>
<td>$(\Box T)$</td>
<td>$T \vdash \Box T$</td>
</tr>
<tr>
<td>$(\Box T)$</td>
<td>$A \vdash B \vdash \Box A \rightarrow \Box B$</td>
</tr>
<tr>
<td>$(\Diamond n[])$</td>
<td>$n[\Diamond A] \vdash \Diamond n[A]$</td>
</tr>
<tr>
<td>$(\Diamond 1)$</td>
<td>$\Diamond A \vdash \Diamond (A \rightarrow B)$</td>
</tr>
<tr>
<td>$(\Diamond \Diamond)$</td>
<td>$\Diamond \Diamond A \vdash \Diamond \Diamond A$</td>
</tr>
</tbody>
</table>

S4, but not S5:  
\[ \neg \text{vld} \; \Diamond A \vdash \Box \Diamond A \quad \neg \text{vld} \; \Diamond A \vdash \Box \Diamond A \]

$(\Diamond \Diamond)$: if somewhere sometime $A$, then sometime somewhere $A$
**Equality**

- **Name equality** can be defined within the logic:

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

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”:

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

\[ Im \triangleq T \triangleright □(obj[] \mid T) \]

\[ Ir \triangleq T \triangleright □\Diamond \neg (obj[] \mid T) \]

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

\[ \vdash □(obj[] \mid T) \]

\[ \vdash \Diamond □(obj[] \mid T) \]

\[ T \mid (T \triangleright □\Diamond \neg (obj[] \mid T)) \]

\[ \vdash □\Diamond \neg (obj[] \mid T) \]

\[ \vdash \neg \Diamond □(obj[] \mid T) \]

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

\[ A \land \neg A \vdash F \]
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:

  $$(\forall n)P \equiv (\forall m)(P\{n\leftarrow m\})$$
  $$(\forall n)0 \equiv 0$$
  $$(\forall n)(\forall m)P \equiv (\forall m)(\forall n)P$$
  $$(\forall n)(P \mid Q) \equiv (\forall n)P \mid Q \text{ if } n \notin \text{fn}(Q)$$

  a.k.a. $$(\forall n)(P \mid (\forall n)Q^{'}) \equiv (\forall n)P \mid (\forall n)Q^{'}$$
  $$(\forall n)(m[P]) \equiv m[(\forall n)P] \text{ if } n \neq m$$

  a private name is as good as another

  scope extrusion

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

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

• \( n \mathcal{R} \mathcal{A} \) is read, informally:
  
  • \textit{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:
  
  • \( n \mathcal{R} n[0] \): reveal a restricted name (say, \( p \)) as \( n \) and check the presence of an empty \( n \) location in the revealed process.

\[
(\forall p)p[0] \models n \mathcal{R} n[0]
\]

because \((\forall p)p[0] \equiv (\forall n)n[0]\) and \( n[0] \models n[0] \)
### Derived Formulas: Revelation

<table>
<thead>
<tr>
<th>$\Diamond n$</th>
<th>$\equiv \neg n \circ \top$</th>
<th>$P \vdash -$ iff $\neg \exists P' \in \Pi. \ P \equiv (\forall n)P'$ iff $n \in \text{fn}(P)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{closed}$</td>
<td>$\equiv \neg \exists x. \Diamond x$</td>
<td>$P \vdash -$ iff $\neg \exists n \in \Lambda. \ n \in \text{fn}(P)$</td>
</tr>
<tr>
<td>$\text{separate}$</td>
<td>$\equiv \neg \exists x. \Diamond x \</td>
<td>\ \Diamond x$</td>
</tr>
</tbody>
</table>

#### Examples:

- $n[] \vdash \Diamond n$
- $(\forall p)p[] \vdash \text{closed}$
- $n[] \ | \ m[] \vdash \text{separate}$
Revelation Rules

- Some mirror properties of restriction:
  \[
  \begin{align*}
  & x \text{®} x \text{®} A \vdash x \text{®} A \\
  & x \text{®} y \text{®} A \vdash y \text{®} x \text{®} A \\
  & x \text{®} (A \mid x \text{®} B) \vdash x \text{®} A \mid x \text{®} B \\
  \end{align*}
  \]

  (scope extrusion)

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

- Some deal with the adjunction:
  \[
  \begin{align*}
  & \eta \text{®} A \vdash B \\
  & \vdash (\neg A) \text{®} x \vdash \neg (A \text{®} x) \\
  & (A \mid B) \text{®} x \vdash A \text{®} x \mid B \text{®} x \\
  & x \text{®} ((A \mid B) \text{®} x) \vdash x \text{®} (A \text{®} x) \mid x \text{®} (B \text{®} x)
  \end{align*}
  \]
### 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>$A \vdash B \quad x \circ A \vdash x \circ B$</td>
</tr>
<tr>
<td>(© ⊗)</td>
<td>$\eta \circ A \vdash B \quad A \vdash B \circ \eta$</td>
</tr>
<tr>
<td>(© ¬)</td>
<td>$(\neg A) \circ x \vdash \neg (A \circ x)$</td>
</tr>
<tr>
<td>(© ▷ F)</td>
<td>$A^F \circ x \vdash A^F$</td>
</tr>
</tbody>
</table>

- idempotency
- commutativity
- $\circ \lor$ distribution
- congruence
- $\circ \circ$ adjunction
- is self-dual
- unsatisfiable
\[
\begin{align*}
\text{(® 0)} & \quad \{ x \text{®} 0 \vdash 0 \} & \text{®/®-0 rules} \\
\text{(⊙ 0)} & \quad \{ 0 \text{⊙} x \vdash 0 \} \\
\text{(® 1)} & \quad \{ x \text{®}(A \mid x \text{®} B) \vdash x \text{®} A \mid x \text{®} B \} & \text{®/®-1 rules} \\
\text{(⊙ 1)} & \quad \{ (A \mid B) \text{⊙} x \vdash A \text{⊙} x \mid B \text{⊙} x \} \\
\text{(® ⊙ 1)} & \quad \{ x \text{®}((A \mid B) \text{⊙} x) \vdash x \text{®}(A \text{⊙} x) \mid x \text{®}(B \text{⊙} x) \} \\
\text{(® n[[]])} & \quad \{ x \text{®} y[A] \vdash y[x \text{®} A] \} & (x \neq y) & \text{®/⊙-n[-] rules} \\
\text{(⊙ n[[]])} & \quad \{ y[A] \text{⊙} x \vdash y[A \text{⊙} x] \} & (x \neq y) \\
\text{(⊙ n[[]])} & \quad \{ x[A] \text{⊙} x \vdash F \}
\end{align*}
\]
Fresh-Name Quantifier

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

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

- Fundamental “freshness” property (Gabbay-Pitts):
  \[ \forall x. \mathcal{A} \iff \exists m \in \Lambda. m \notin \text{fn}(P, \mathcal{A}) \land P \models \mathcal{A}\{x \leftarrow m\} \]
  
  iff \[ \forall m \in \Lambda. m \notin \text{fn}(P, \mathcal{A}) \Rightarrow P \models \mathcal{A}\{x \leftarrow m\} \]
  
  because any fresh name is as good as any other.
• Very nice logical properties:
  - $\forall x. \mathcal{A} \vdash \forall x. \mathcal{A} \vdash \exists x. \mathcal{A}$
  - $\neg \forall x. \mathcal{A} \vdash \forall x. \neg \mathcal{A}$
• $\forall x. (\mathcal{A} \lor \mathcal{B}) \vdash (\forall x. \mathcal{A}) \lor (\forall x. \mathcal{B})$  (hint: (GP) $\exists$ for $\Rightarrow$, $\forall$ for $\Leftarrow$)
• $\Diamond \forall x. \mathcal{A} \vdash \forall x. \Diamond \mathcal{A}$
Hidden-Name Quantifier

\[ Hx.\mathcal{A} \equiv \forall x. x \oplus \mathcal{A} \]

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

• Example: \( Hx.x[] = \forall x. x \oplus x[] \)

• “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 \( x \)”

• \( (\forall n)n[] \models Hx.x[] \) because \( (\forall n)n[] \models \forall x. x \oplus x[] \)

because \( (\forall n)n[] \models n \oplus n[] \) (where \( n \notin fn((\forall n)n[]) \)).

• Counterexamples:

• \( (\forall m)m[] \not\models Hx.n[] \) \hspace{1cm} (N.B.: this holds for \( Hx.\mathcal{A} \equiv \exists x. x \oplus \mathcal{A} \) !)

• \( (\forall n)n[] \mid (\forall n)n[] \not\models Hx.(x[] \mid x[]) \)

• \( (\forall n)(n[] \mid n[]) \not\models Hx.x[] \mid Hx.x[] \)
Consider:
\[ \forall x. x \mathbb{R} (A \mid x \mathbb{B}) \]
\[ \models \forall x. (x \mathbb{R} A \mid x \mathbb{R} B) \]
\[ \models (\forall x. x \mathbb{R} A) \mid (\forall x. x \mathbb{R} B) \]

That is:
\[ Hx. (A \mid x \mathbb{B}) \models Hx. A \mid Hx. B \]

Hence, the scope extrusion rule for \( H \) still uses \( \mathbb{R} \).

No matter what one choses as primitives, we have explored interesting connections between these operators. (\( \mathbb{N} + \mathbb{R} \) and \( H + \mathbb{C} \) 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”.

\[ Hx.(n[\odot x] \mid m[\odot x]) \]

- \( P \vdash Hx.(n[\odot x] \mid m[\odot x]) \)

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

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

- A protocol establishing a shared key should satisfy:

\[ \Box Hx.(n[\odot x] \mid m[\odot x]) \]
From Logic back to Types

- 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 $P : A$ means that process $P$ may have “effects” $A$, 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:
    \[ \Gamma \vdash P : A, \quad \Gamma \vdash Q : B \implies \Gamma \vdash P \parallel Q : A \parallel B \]
  - Effects may be hidden:
    \[ \Gamma, n : A \vdash P : B(x \leftarrow n) \implies \Gamma \vdash (vn : A)P : Hx : A. B \]
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 $E, n: \text{Amb}^\bullet[S] \vdash P : T$ (a typing judgment asserting that no ambient called $n$ can ever be opened in $P$), then:
      $$P \not\models \Box(\Diamond an \ n \Rightarrow \Box \Diamond an \ n)$$
  - Expressing Immobility
    - If $E, p: \text{Amb}^\bullet[S], q: \text{Amb}^\bullet[S'] \vdash P : T$ (a typing judgment asserting that no ambient called $q$ can ever move within $P$), then:
      $$P \not\models \Box(\Diamond (p \text{ parents } q) \Rightarrow \Box \Diamond (p \text{ parents } q))$$
      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 \( \{ (\mathcal{A} \mid B) \land 0 \models \mathcal{A} \} \) 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 \( \{ (\mathcal{A} \mid B) \land 0 \models \mathcal{A} \} \) from the axioms in the slides. (My) proof uses the decomposition axiom, \( (\mid \mid ) \).
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 $\exists x. x^\mathbb{R}A$ and $\forall x. x^\mathbb{R}A$. This is even derivable within the logic:

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

• It implies:

$$P \models A \Rightarrow (\forall n)P \models Hx.(A\{n\leftarrow x\})$$

$$P \models Hx.(A\{n\leftarrow x\}) \land n \notin \text{fn}(P) \Rightarrow P \models n^\mathbb{R}A$$

$$P \models n^\mathbb{R}A \Rightarrow P \models Hx.(A\{n\leftarrow x\})$$
A Surprising Property

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

- **Ex.:** \[ \text{Hx.}(\neg 0 \mid \neg 0) \not\models \neg 0 \mid \neg 0 \]

  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:

  \[ (\forall n)(n[] \mid n[]) \models \forall x.x\mathcal{R}(\neg 0 \mid \neg 0) \quad \text{but:} \]

  \[ (\forall n)(n[] \mid n[]) \not\models \neg 0 \mid \neg 0. \]

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

- However, \( \text{Hx.}0 \vdash 0. \)

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

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

\[
\begin{align*}
\triangle n \\
\Downarrow P \\
\vdash Hx.\mathcal{A} \quad \text{if} \\
\Downarrow P \\
\vdash \mathcal{A}\{x\leftarrow n\}
\end{align*}
\]

- Design decision: how to define $Hx.\mathcal{A}$, keeping in mind that “freshness” may spill into the logic?
  - *The Obvious Thing*: extend the syntax with $Hx.\mathcal{A}$ and define it directly.
  - *Luis Caires*: Extend the syntax with $Hx.\mathcal{A}$ and add signatures to keep track of free names, to enforce the side condition $n \notin fn(\mathcal{A})$: $\Sigma \cdot P \vdash \Sigma \cdot \mathcal{A}$.
  - *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: $Hx.\mathcal{A} \triangleq \forall x. x \mathcal{A}$. 
The Decomposition Operator

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

\[ \mathcal{A} \parallel \mathcal{B} \triangleq \neg(\neg\mathcal{A} \| \neg\mathcal{B}) \quad P \vdash \text{iff} \quad \forall P', P'' \in \Pi. \ P \equiv P' \| P'' \Rightarrow P' \vdash \mathcal{A} \lor P'' \vdash \mathcal{B} \]

\[ \mathcal{A}^\forall \triangleq \mathcal{A} \parallel \text{F} \quad P \vdash \text{iff} \quad \forall P', P'' \in \Pi. \ P \equiv P' \| P'' \Rightarrow P' \vdash \mathcal{A} \]

\[ \mathcal{A}^\exists \triangleq \mathcal{A} \| \text{T} \quad P \vdash \text{iff} \quad \exists P', P'' \in \Pi. \ P \equiv P' \| P'' \land P' \vdash \mathcal{A} \]

\[ \mathcal{A} \parallel \mathcal{B} \quad \text{for every partition, one piece satisfies } \mathcal{A} \quad \text{or the other piece satisfies } \mathcal{B} \]

\[ \mathcal{A}^\forall \leftrightarrow \neg((\neg\mathcal{A})^\exists) \quad \text{every component satisfies } \mathcal{A} \]

\[ \mathcal{A}^\exists \leftrightarrow \neg((\neg\mathcal{A})^\forall) \quad \text{some component satisfies } \mathcal{A} \]

Examples:

\[ (p[T] \Rightarrow p[q[T]^\exists])^\forall \quad \text{every } p \text{ has a } q \text{ child} \]

\[ (p[T] \Rightarrow p[q[T] \| (\neg q[T])^\forall])^\forall \quad \text{every } p \text{ has a unique } q \text{ child} \]
The Decomposition Axiom

\[ (1 \Vert) \implies (\mathcal{A}' \parallel \mathcal{A}'') \vdash (\mathcal{A}' \parallel \mathcal{B}'') \lor (\mathcal{B}' \parallel \mathcal{A}'') \lor (\neg \mathcal{B}' \parallel \neg \mathcal{B}'') \]

- **Alternative formulations and special cases:**

  \[ (\mathcal{A}' \parallel \mathcal{A}'') \land (\mathcal{B}' \parallel \mathcal{B}'') \vdash (\mathcal{A}' \parallel \mathcal{B}'') \lor (\mathcal{B}' \parallel \mathcal{A}'') \]

  “If \( P \) has a partition into pieces that satisfy \( \mathcal{A}' \) and \( \mathcal{A}'' \), and every partition has one piece that satisfies \( \mathcal{B}' \) or the other that satisfies \( \mathcal{B}'' \), then either \( P \) has a partition into pieces that satisfy \( \mathcal{A}' \) and \( \mathcal{B}'' \), or it has a partition into pieces that satisfy \( \mathcal{B}' \) and \( \mathcal{A}'' \).”

  \[ \neg(\mathcal{A} \parallel \mathcal{B}) \vdash (\mathcal{A} \parallel \mathcal{F}) \implies (\mathcal{F} \parallel \neg \mathcal{B}) \]

  “If \( P \) has no partition into pieces that satisfy \( \mathcal{A} \) and \( \mathcal{B} \), but \( P \) has a piece that satisfies \( \mathcal{A} \), then \( P \) has a piece that does not satisfy \( \mathcal{B} \).”

  \[ \neg(\mathcal{F} \parallel \mathcal{B}) \vdash \mathcal{F} \parallel \neg \mathcal{B} \]

  \[ \neg(\mathcal{A} \parallel \mathcal{B}) \vdash (\neg \mathcal{A} \parallel \mathcal{F}) \lor (\mathcal{F} \parallel \neg \mathcal{B}) \]
Logical Adjunctions

- This is a logic with multiple logical adjunctions (4 of them!):
  \[ \land / \Rightarrow \quad \text{(classical)} \]
  \[ \mathcal{A} \land C \vdash B \quad \text{iff} \quad \mathcal{A} \vdash C \Rightarrow B \]
- \[ \mid / \triangleright \quad \text{(linear, } \otimes / \twoheadrightarrow \text{)} \]
  \[ \mathcal{A} \mid C \vdash B \quad \text{iff} \quad \mathcal{A} \vdash C \triangleright B \]
- \[ n[-] / \neg \uparrow n \]
  \[ n[\mathcal{A}] \vdash B \quad \text{iff} \quad \mathcal{A} \vdash B \uparrow n \]
- \[ n\otimes - / \neg \odot n \]
  \[ n\otimes \mathcal{A} \vdash B \quad \text{iff} \quad \mathcal{A} \vdash B \otimes n \]
- 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” Sequent

• 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 $\triangleright$.

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

**location a**

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

**location b**

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

send M from a to b

receive n; do P

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

- (exit) → \[ a[] \]
- (enter) → \[ a[] \]
- (open) → \[ a[] \]
- (read) → \[ a[] \]
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:

\[
\begin{align*}
1_{\text{MILL}} \cong 0 \\
A \otimes_{\text{MILL}} B \cong A \mid B \\
A \multimap_{\text{MILL}} B \cong A \triangleright B
\end{align*}
\]

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

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

\[
\begin{align*}
A + B & \triangleq A \lor B \\
A \& B & \triangleq A \land B \\
A \otimes B & \triangleq A \mid B \\
A \rightarrow B & \triangleq A \| B \\
!A & \triangleq 0 \land (0 \Rightarrow A)^{-F}
\end{align*}
\]

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

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

• Some discrepancies: \( \bot_{\text{ILL}} = 0_{\text{ILL}} \); the additives distribute; !A is not “replication”; !A→B is not so interesting; \( A^\top / A^0 \) 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 $\mathcal{A} \vdash \mathcal{B}$ is defined just like intensional conjunction, and $\mathcal{A} \rhd \mathcal{B}$ is defined just like relevant implication in that semantics.

• Except:
  • We do not have contraction. This does not make sense in process calculi, because $P \vdash P \neq P$. 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 *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 *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
  
  $\langle S \in \text{Set}, \leq \in S^2 \rightarrow \text{Bool}, \lor \in \mathcal{P}(S) \rightarrow S, \otimes \in S^2 \rightarrow S, 1 \in S \rangle$ such that:
  
  $\leq, \lor$ is a complete join semilattice
  
  $\otimes, 1$ is a commutative monoid

  $p \otimes \lor Q = \lor \{p \otimes q \mid q \in Q\}$

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

  $\llbracket A \oplus B \rrbracket \triangleq \lor \{\llbracket A \rrbracket, \llbracket B \rrbracket\}$ \hspace{1cm} $\llbracket 1_{\text{ILL}} \rrbracket \triangleq 1$

  $\llbracket A \& B \rrbracket \triangleq \lor \{C \mid C \leq \llbracket A \rrbracket \land C \leq \llbracket B \rrbracket\}$ \hspace{1cm} $\llbracket \bot_{\text{ILL}} \rrbracket \triangleq$ any element of $S$

  $\llbracket A \otimes B \rrbracket \triangleq \llbracket A \rrbracket \otimes \llbracket B \rrbracket$ \hspace{1cm} $\llbracket \top_{\text{ILL}} \rrbracket \triangleq \lor S$

  $\llbracket A \rightarrow B \rrbracket \triangleq \lor \{C \mid C \otimes \llbracket A \rrbracket \leq \llbracket B \rrbracket\}$ \hspace{1cm} $\llbracket 0_{\text{ILL}} \rrbracket \triangleq \lor \emptyset$

  $\llbracket !A \rrbracket \triangleq \forall X. \llbracket 1 \& A \& X \otimes X \rrbracket$ where $\forall X. A\{X\} \triangleq \lor \{C \mid C \leq A\{C\}\}$

  $\text{vld}_{\text{ILL}}(A_1, \ldots, A_n \vdash_{\text{ILL}} B)_Q \triangleq \llbracket A_1 \rrbracket_Q \otimes_Q \ldots \otimes_Q \llbracket A_n \rrbracket_Q \leq_Q \llbracket B \rrbracket_Q$
The Process Quantale

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

  $$\Theta \triangleq < \Phi, \subseteq, \cup, \otimes, 1>$$

  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 $\Theta$:

  $$\text{vld}_{\text{ILL}}(A_1, ..., A_n \vdash_{\text{ILL}} B)_\Theta$$

  $$\iff [A_1] \otimes_\Theta ... \otimes_\Theta [A_n] \subseteq [B]$$

  $$\iff [A_1 \mid ... \mid A_n] \subseteq [B]$$

  $$\iff (\Pi - [A_1 \mid ... \mid A_n]) \cup [B] = \Pi$$

  $$\iff [A_1 \mid ... \mid A_n \Rightarrow B] = \Pi$$
• Semantic domain: $\Theta$

$$\Pi \triangleq \text{the set of process expressions}$$

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

$$\Phi \triangleq \{ C^= \mid C \subseteq \Pi \}$$

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

• Spatial operators over $\Theta$:

$$1 \triangleq \{ \emptyset \}^= $$

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

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

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

\[ n\oplus C \triangleq \{ (\forall n)P \mid P \in C \}\equiv \]

- 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).

- \(n\oplus C\) 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 \(C\).

- \(Ex.: \ n\oplus n[1]\): reveal a private name (say, \(p\)) as \(n\) and check that there is an empty \(n\) ambient in the revealed process.
  
  \((\forall p)p[0] \in n\oplus n[1]\)

  because \((\forall p)p[0] \equiv (\forall n)n[0]\) and \(n[0] \in n[1]\)
More examples of $n \mathbin{\circ} C \triangleq \{ \forall n) P \mid P \in C \}$:

- $0 \in n \mathbin{\circ} 1$ because $0 \equiv (\forall n)0$ and $0 \in 1$
- $m[0] \in n \mathbin{\circ} \Pi$ because $m[0] \equiv (\forall n)m[0]$ and $m[0] \in \Pi$
- $n[0] \notin n \mathbin{\circ} \Pi$ because $n[0] \not\equiv (\forall n)$...

Therefore, $n \mathbin{\circ} C$ 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: $\neg n \mathbin{\circ} \mathcal{A} \land \Diamond n \mathbin{\circ} \mathcal{A}$
# Semantics of the Logic

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>([T])</td>
<td>(\equiv \Pi)</td>
</tr>
<tr>
<td>(\neg A)</td>
<td>(\equiv \Pi - [A])</td>
</tr>
<tr>
<td>(A \lor B)</td>
<td>(\equiv [A] \cup [B])</td>
</tr>
<tr>
<td>([0])</td>
<td>(\equiv 1)</td>
</tr>
<tr>
<td>(n[A])</td>
<td>(\equiv n[[A]])</td>
</tr>
<tr>
<td>(A@n)</td>
<td>(\equiv \bigcup{C \in \Theta \mid n[C] \subseteq [A]})</td>
</tr>
<tr>
<td>(A \mid B)</td>
<td>(\equiv [A] \times [B])</td>
</tr>
<tr>
<td>(A&gt;B)</td>
<td>(\equiv \bigcup{C \in \Theta \mid C@\Theta[A] \subseteq [B]})</td>
</tr>
<tr>
<td>(n\circ A)</td>
<td>(\equiv n\circ\Theta[A])</td>
</tr>
<tr>
<td>(A@n)</td>
<td>(\equiv \bigcup{C \in \Theta \mid n@\circ\Theta C \subseteq [A]})</td>
</tr>
<tr>
<td>(\exists A)</td>
<td>(\equiv {P \in \Pi \mid \exists P' \in \Pi. P \downarrow^* P' \land P' \in [A]})</td>
</tr>
<tr>
<td>(\forall A)</td>
<td>(\equiv {P \in \Pi \mid \exists P' \in \Pi. P \rightarrow^* P' \land P' \in [A]})</td>
</tr>
<tr>
<td>(\forall x. A)</td>
<td>(\equiv \bigcap_{m \in \Lambda} [A{x \leftarrow m}])</td>
</tr>
</tbody>
</table>

\(P \downarrow P'\) 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:

\[ \forall \alpha \in \Phi. [\alpha] \in \Theta \]
• 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.