Logics for Mobility

Luca Cardelli
Andy Gordon

Microsoft Research

Lausanne 2000-10-24
Simple Properties of Mobile Computation

- We have been looking for ways to express properties of mobile computations, E.g.:
  - “Here today, gone tomorrow.”
  - “Eventually the agent crosses the firewall.”
  - “Every agent carries a suitcase.”
  - “Somewhere there is a virus.”
  - “There is always at most one entity called $n$ here.”

- As with properties of ordinary concurrent computations, formalization options include:
  - Type systems (limited).
  - Equational reasoning (hard).
  - Reasoning on traces (ugly).
  - Reasoning via modal/temporal logics (a popular compromise).
Harder Properties

• Moreover, 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.)
- Find ways of expressing properties of dynamically changing structures of locations.
  - Previous work [POPL’00].
- Find ways of talking about hidden names. We split it into two logical tasks:
  - Find ways of quantifying over fresh names. We adopt a recent solution [Gabbay-Pitts].
  - Find ways of revealing 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 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**

- $0$ (void)
- $n[P]$ (location)
- $P \mid Q$ (composition)

**Formulas**

- $0$ (there is nothing here)
- $n[\mathcal{A}]$ (there is one thing here)
- $\mathcal{A} \mid \mathcal{B}$ (there are two things here)

**Trees**

- (void)
- (location)
- (composition)
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[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.

\[ a[c[out \ a. \ in \ b. \ P]] \ | \ b[0] \]
Mobility

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

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

\[ a \]  
\[ c \]  
\[ b \]  

\[ a[Q] \]  
\[ c[{\it in } b. P] \]  
\[ b[R] \]
Mobility

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

```
\begin{align*}
a[Q] & \quad \mid b[R \mid c[P]]
\end{align*}
```
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, …

Now
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.
### Ambient Calculus

- **Processes**
  - \( (\forall n)P \): restriction
  - 0: inactivity
  - \( P \mid P' \): parallel
  - \( M[P] \): ambient
  - !P: replication
  - \( M.P \): exercise a capability
  - \((n).P\): input locally, bind to \(n\)
  - \(\langle M\rangle\): output locally (async)

- **Messages**
  - \( M \)::=
    - in \(M\): entry capability
    - out \(M\): exit capability
    - open \(M\): open capability
    - \(\varepsilon\): empty path
    - \(M.M'\): composite path

\[
\begin{align*}
n[] &\triangleq n[0] \\
M &\triangleq M.0 \quad \text{(where appropriate)}
\end{align*}
\]
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:
    $$P \equiv P', P' \rightarrow^* Q', Q' \equiv Q \quad \Rightarrow \quad P \rightarrow^* Q$$
**Restriction (much as in the \(\pi\)-calculus)**

- \((\forall n)P\)
  - “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:

\[
(\forall n)P \equiv (\forall m)(P[n \leftarrow m])
\]
\[
(\forall n)0 \equiv 0
\]
\[
(\forall n)(\forall m)P \equiv (\forall m)(\forall n)P
\]
\[
(\forall n)(P \| Q) \equiv P \| (\forall n)Q \quad \text{if } n \notin fn(P)
\]
\[
(\forall n)(m[P]) \equiv m[(\forall n)P] \quad \text{if } n \neq m
\]

- Uses or restriction:
  - Initially to represent private channels.
  - Later, to represent private names of any kind:
    - Channels, Locations, Nonces, Cryptokeys, …
Modal Logics

• 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

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

- \( T \) true
- \( \neg \mathcal{A} \) negation
- \( \mathcal{A} \lor \mathcal{A}' \) disjunction
- 0 void
- \( \eta[\mathcal{A}] \) location
- \( \mathcal{A} \diamond \mathcal{A}' \) composition
- \( \eta \Diamond \mathcal{A} \) revelation
- \( \Diamond \mathcal{A} \) somewhere modality
- \( \forall x. \mathcal{A} \) universal quantification over names
- \( \mathcal{A} @ \eta \) location adjunct
- \( \mathcal{A} \oplus \eta \) revelation adjunct
- \( \mathcal{A} \oplus \eta \) composition adjunct
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 \( q \)
### Satisfaction Relation

<table>
<thead>
<tr>
<th>$P \models T$</th>
<th>$\triangleleft \quad \neg P \models \mathcal{A}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \models \neg \mathcal{A}$</td>
<td>$\triangleleft \quad P \models \mathcal{A} \lor P \models \mathcal{B}$</td>
</tr>
<tr>
<td>$P \models \mathcal{A} \lor \mathcal{B}$</td>
<td>$\triangleleft \quad P \models \mathcal{A} \lor P \models \mathcal{B}$</td>
</tr>
<tr>
<td>$P \models 0$</td>
<td>$\triangleleft \quad P \models 0$</td>
</tr>
<tr>
<td>$P \models n[\mathcal{A}]$</td>
<td>$\triangleleft \quad \exists P' \epsilon \Pi. P \equiv n[P'] \land P' \models \mathcal{A}$</td>
</tr>
<tr>
<td>$P \models \mathcal{A}@n$</td>
<td>$\triangleleft \quad n[P] \models \mathcal{A}$</td>
</tr>
<tr>
<td>$P \models \mathcal{A} \mid \mathcal{B}$</td>
<td>$\triangleleft \quad \exists P', P'' \epsilon \Pi. P \equiv P' \land P'' \land P' \models \mathcal{A} \land P'' \models \mathcal{B}$</td>
</tr>
<tr>
<td>$P \models \mathcal{A} \triangleright \mathcal{B}$</td>
<td>$\triangleleft \quad \forall P' \epsilon \Pi. P' \models \mathcal{A} \Rightarrow P \mid P' \models \mathcal{B}$</td>
</tr>
<tr>
<td>$P \models n@\mathcal{A}$</td>
<td>$\triangleleft \quad \exists P' \epsilon \Pi. P \equiv (\forall n)P' \land P' \models \mathcal{A}$</td>
</tr>
<tr>
<td>$P \models \mathcal{A} \bowtie n$</td>
<td>$\triangleleft \quad (\forall n)P \models \mathcal{A}$</td>
</tr>
<tr>
<td>$P \models \mathcal{A}$</td>
<td>$\triangleleft \quad \exists P' \epsilon \Pi. P \downarrow^* P' \land P' \models \mathcal{A}$</td>
</tr>
<tr>
<td>$P \models \mathcal{A}$</td>
<td>$\triangleleft \quad \exists P' \epsilon \Pi. P \rightarrow^* P' \land P' \models \mathcal{A}$</td>
</tr>
<tr>
<td>$P \models \forall x. \mathcal{A}$</td>
<td>$\triangleleft \quad \forall m \epsilon \Lambda. P \models \mathcal{A}{x\leftarrow m}$</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

- 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.
Basic Tree Formulas

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

- \textbf{0}: there is no structure here now.
- \textbf{n[\mathcal{A}]}: there is a location \textit{n} with contents satisfying \textit{\mathcal{A}}.
- \textbf{\mathcal{A} \mid \mathcal{B}}: there are two structures satisfying \textit{\mathcal{A}} and \textit{\mathcal{B}}.
- \textbf{\mathcal{A} @ n}: when the current structure is placed in a location \textit{n}, the resulting structure satisfies \textit{\mathcal{A}}.
- \textbf{\mathcal{A} \triangleright \mathcal{B}}: when the current structure is composed with one satisfying \textit{\mathcal{A}}, the resulting structures satisfies \textit{\mathcal{B}}.
Satisfaction for Basic Trees

- \( \models 0 \)

- \( \models n[\mathcal{A}] \) if \( \models \mathcal{A} \)

- \( \models \mathcal{A} \mid \mathcal{B} \) if \( \models \mathcal{A} \) and \( \models \mathcal{B} \)

- \( \models \mathcal{A} @ n \) if \( \models \mathcal{A} \)

- \( \models \mathcal{A} \triangleright \mathcal{B} \) if for all \( \models \mathcal{A} \) we have \( \models \mathcal{B} \)

Ambient Logic - Lausanne 22
Satisfaction for Somewhere/Sometime

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

\[ P \models \diamond \mathcal{A} \quad \text{if} \quad P \rightarrow^* Q \quad \text{and} \quad Q \models \mathcal{A} \]
Satisfaction for Revelation

- Trees with hidden labels:

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

\[ P \overset{n}{=} n \bowtie \mathcal{A} \quad \text{if} \quad P \overset{\mathcal{A}}{=} \]

\[ P \overset{\mathcal{A} \uplus n}{=} \quad \text{if} \quad P \overset{\mathcal{A}}{=} \]
Revelation

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

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

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

\[
(\forall p)p[0] \models n^{\mathbb{A}}n[0]
\]

because \((\forall p)p[0] \equiv (\forall n)n[0]\) and \( n[0] \models n[0] \)
• $0 \models n^\circ 0$ because $0 \equiv (\forall n)0$ and $0 \models 0$
• $m[0] \models n^\circ T$ because $m[0] \equiv (\forall n)m[0]$ and $m[0] \models T$
• $n[0] \not\equiv n^\circ T$ because $n[0] \not\equiv (\forall n)...

Therefore, the set of processes satisfying $n^\circ A$ 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: $\neg n^\circ A \wedge \Diamond n^\circ A$
## Derived Formulas

<table>
<thead>
<tr>
<th>Formula</th>
<th>Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F )</td>
<td>( \equiv \neg T )</td>
</tr>
<tr>
<td>( \mathcal{A} \rightarrow B )</td>
<td>( \equiv \neg \mathcal{A} \lor B )</td>
</tr>
<tr>
<td>( \mathcal{A} \land B )</td>
<td>( \equiv \neg (\neg \mathcal{A} \lor \neg B) )</td>
</tr>
<tr>
<td>( \exists x. \mathcal{A} )</td>
<td>( \equiv \neg \forall x. \neg \mathcal{A} )</td>
</tr>
<tr>
<td>( \Box \mathcal{A} )</td>
<td>( \equiv \neg \Diamond \neg \mathcal{A} )</td>
</tr>
<tr>
<td>( \mathcal{A} \land F )</td>
<td>( \equiv \mathcal{A} \rightarrow F )</td>
</tr>
<tr>
<td>( \mathcal{A} ) valid</td>
<td>( P \vdash \iff \forall P' \in \Pi. P' \vdash \mathcal{A} \Rightarrow P \mid P' \vdash F )</td>
</tr>
<tr>
<td>( \mathcal{A} ) satisfiable</td>
<td>( P \vdash \iff \exists P' \in \Pi. P' \vdash \mathcal{A} )</td>
</tr>
</tbody>
</table>
**Derived Formulas: Revelation**

<table>
<thead>
<tr>
<th> </th>
<th> </th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Box n$</td>
<td>$\triangleq \neg n \supset T$</td>
</tr>
<tr>
<td>$P \models -$</td>
<td>iff $\neg \exists P' \in \Pi. P \equiv (\forall n)P'$</td>
</tr>
<tr>
<td>$\text{iff } n \in fn(P)$</td>
<td></td>
</tr>
<tr>
<td>$\text{closed }$</td>
<td>$\triangleq \neg \exists x. \Box x$</td>
</tr>
<tr>
<td>$P \models -$</td>
<td>iff $\neg \exists n \in \Lambda. n \in fn(P)$</td>
</tr>
<tr>
<td>$\text{separate }$</td>
<td>$\triangleq \neg \exists x. \Box x \mid \Box x$</td>
</tr>
<tr>
<td>$P \models -$</td>
<td>iff $\neg \exists n \in \Lambda. P' \in \Pi, P'' \in \Pi.$</td>
</tr>
<tr>
<td>$P \equiv P' \mid P'' \land n \in fn(P') \land n \in fn(P'')$</td>
<td></td>
</tr>
</tbody>
</table>

**Examples:**

- $n[] \models \Box n$
- $(\forall p)p[] \models \text{closed}$
- $n[] \mid m[] \models \text{separate}$
From Satisfaction to (Propositional) Logic

• Propositional validity

\[ \text{vld} \ A \triangleq \ \forall P \in \Pi. \ P \vdash A \]

\( A \) (closed) is valid

• Sequents

\[ A \vdash B \triangleq \ \forall P \in \Pi. \ P \models A \Rightarrow P \models B \]

• Rules

\[ A_1 \vdash B_1; \ldots; A_n \vdash B_n \ \vdash A \vdash B \triangleq \ (n \geq 0) \]

\[ A_1 \vdash B_1 \land \ldots \land A_n \vdash B_n \Rightarrow A \vdash B \]

(N.B.: all the rules shown later are validated accordingly.)

• Conventions:

- \( \leftrightarrow \) means \( \vdash \) in both directions

\( \{ \} \) means \( \{ \} \) in both directions
• Logical axioms and rules.
  • Rules of propositional logic (standard).
  • Rules of location and composition
    \[ \mathcal{A} \vdash C \rightarrow B \quad \iff \quad \mathcal{A} \vdash C \triangleright B \quad \dashv \vdash \text{adjunction} \]
  • Rules of revelation
    \[ \eta \mathcal{A} \vdash B \quad \iff \quad \mathcal{A} \vdash B \ominus \eta \]
    \[ \{ (\neg \mathcal{A}) \ominus x \vdash \neg (\mathcal{A} \ominus x) \]  \( \ominus \) 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.
Ex: Immovable Object vs. Irresistible Force

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

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

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

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

\[ A \land \neg A \vdash F \]
Example: Thief!

- A *shopper* is likely to pull out a wallet. A *thief* is likely to grab it.

\[
\text{Shopper} \triangleq \\
\text{Person}[\text{Wallet}[$] \mid \text{T}] \land \\
\Diamond (\text{Person}[\text{NoWallet}] \mid \text{Wallet}[$])
\]

\[
\text{NoWallet} \triangleq \neg (\text{Wallet}[$] \mid \text{T})
\]

\[
\text{Thief} \triangleq \text{Wallet}[$] \triangleright \Diamond \text{NoWallet}
\]

- By simple logical deductions involving laws of \(\triangleright\) and \(\Diamond\):

\[
\text{Shopper} \mid \text{Thief} \Rightarrow \\
\left( \text{Person} [\text{Wallet}[$] \mid \text{T}] \mid \text{Thief} \right) \land \\
\Diamond (\text{Person}[\text{NoWallet}] \mid \text{NoWallet})
\]
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):
  \[ \forall x.\mathcal{A} \iff \exists x. x\#(\text{fn}(\mathcal{A})-\{x\}) \land x \otimes T \land \mathcal{A} \]

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

because *any fresh name as as good as any other.*

- **Very nice properties:**
  - \[ \forall x.\mathcal{A} \Rightarrow \forall x.\mathcal{A} \Rightarrow \exists x.\mathcal{A} \]
    \[ \quad \neg \forall x.\mathcal{A} \iff \forall x. \neg \mathcal{A} \]
  - \[ \forall x. (\mathcal{A} \mid \mathcal{B}) \iff (\forall x.\mathcal{A}) \mid (\forall x.\mathcal{B}) \]
  - \[ \Diamond \forall x.\mathcal{A} \iff \forall x. \Diamond \mathcal{A} \]

because any fresh name as as good as any other.
### Hidden-Name Quantifier

\[(\forall x)A \iff \forall x.x \otimes A\]

- **Example:** \((\forall x)x[T] = \forall x.x \otimes x[T]\)
  - “for hidden \(x\), we find a location called \(x\)” = “for fresh \(x\), we reveal a hidden name as \(x\), then we find a location called \(x\)”
  - \((\forall n)n[] \models (\forall x)x[T]\) because \((\forall n)n[] \models \forall x.x \otimes x[T]\) because \((\forall n)n[] \models n \otimes n[T]\) (where \(n \notin fn((\forall n)n[])\)).

- **Other examples**
  - \((\forall m)m[] \models (\forall x)n[]\)
  - \((\forall n)n[] \models (\forall n)n[] \not\equiv (\forall x)(x[] | x[])\)
  - \((\forall n)(n[] | n[]) \not\equiv (\forall x)x[] | (\forall x)x[]\)
A Good Property

• A property not shared by other candidate definitions (it is even derivable within the logic):

\[(\forall x)(\mathcal{A}\{n\leftarrow x\}) \land n^\perp \mathbf{T} \vdash n^\perp \mathcal{A}\quad \text{where } x \notin \text{fv}(\mathcal{A})\]

It implies:

\[P \vdash \mathcal{A} \Rightarrow (\forall n)P \vdash (\forall x)(\mathcal{A}\{n\leftarrow x\})\]

\[P \vdash n^\perp \mathcal{A} \Rightarrow P \vdash (\forall x)(\mathcal{A}\{n\leftarrow x\})\]

\[P \vdash (\forall x)(\mathcal{A}\{n\leftarrow x\}) \land n \notin \text{fn}(P) \Rightarrow P \vdash n^\perp \mathcal{A}\]
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”.

$$\forall x \ (n[\odot x] \ | \ m[\odot x])$$

• $P \Vdash (\forall x \ (n[\odot x] \ | \ m[\odot x]))$

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

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

• A protocol establishing a shared key should satisfy:

$$\Diamond (\forall x \ (n[\odot x] \ | \ m[\odot x]))$$
Applications

- Verifying security+mobility protocols.
- 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).