A Predicate μ-Calculus for Mobile Processes

Huimin Lin

Laboratory for Computer Science

Institute of Software, Chinese Academy of Sciences

Mobile computing: processes may involve in time as well as in *space*.

Formal models for mobile computing:

π-Calculus, Ambient Calculus, ...

Describing temporal properties of concurrent processes:

CTL, LTL, μ-calculus, ...

What about spatial properties? – *Spatial Logics*

Spatial logics for Mobile Ambients was first proposed by Gordon and Cardelli in POPL'2000

Anytime, Anywhere: Modal logics for mobile ambients

But the logics do not have *recursion*.

The purpose of this talk:

1. Extending ambient logic with fixpoints, resulting in a *predicate* μ-calculus

2. Developing a model checking algorithm for the logic

Related Work

Luis Caires and Luca Cardeli. A Spatial Logic for Concurrency. [Information & Computation, 2003]

For the asynchronous π-calculus
First-order
Fixpoint formulas based on *propositional* variables
Difficulty in defining free names — need supports/" property sets"

W. Charatonik, S. Dal Zilio, A. Gordon, S. Mukhopadhyay and J.-M. Talbot. Model Checking Mobile Ambients. [TCS'2003]

Decidability and complexity results for various fragments of ambient calculus and ambient logics (without fixpoints).

W. Charatonik, A. Gordon and J.-M. Talbot. Finite-Control Mobile Ambients. [ESOP'2002]

Introducing *finite-control* ambients
A model checking algorithm for an ambient logic without fix-points.

Outline of talk:

1. A brief review of Ambient Calculus

2. A predicate μ-calculus

3. A model checking algorithm

The Ambient Calculus

Names: $m,\ n, \ldots \in \mathcal{N}$ (for *locations*)

Capabilities:

$$\alpha ::= \ \text{in}\ n \quad | \quad \text{out}\ n \quad | \quad \text{open}\ n$$

Porcesses:

$$P ::= \ \mathbf{0} \quad | \quad n[P] \quad | \quad \alpha.P \quad | \quad P \mid Q \quad | \quad (\nu n)P \quad |$$

$$(n).P \quad | \quad \langle n \rangle \quad | \quad X \quad | \quad \mathbf{fix} X.P$$

Ambients are located processes with boundaries, organised in a tree-like structure.

An ambient may enter into or exit from other ambients by performing *capability* actions.

Examples

Entry: $n[\mathsf{in}\ m.P \mid Q] \mid m[R] \to m[n[P \mid Q] \mid R]$

Exit: $m[n[\mathsf{out}\ m.P \mid Q] \mid R] \to n[P \mid Q] \mid m[R]$

Open: $n[P \mid Q] \mid \mathsf{open}\ n.R \to P \mid Q \mid R$

Structural congruence

Identifying processes which can not behave differently.

$$
\begin{array}{l}
P \mid \mathbf{0} \equiv P \\
P \mid Q \equiv Q \mid P \\
P \mid (Q \mid R) \equiv (P \mid Q) \mid R \\
(\nu n)\mathbf{0} \equiv \mathbf{0} \\
n \notin \mathit{fn}(P) \Rightarrow (\nu n)(P \mid Q) \equiv P \mid (\nu n)Q \\
(\nu n)(\nu m)P \equiv (\nu m)(\nu n)P \\
n \neq m \Rightarrow (\nu n)m[P] \equiv m[(\nu n)P] \\
\ldots
\end{array}
$$

Reduction rules

$$\text{RedIn}\,\frac{}{n[\mathsf{in}\ m.P \mid Q] \mid m[R] \to m[n[P \mid Q] \mid R]}$$

$$\text{RedOut}\,\frac{}{m[n[\mathsf{out}\ m.P \mid Q] \mid R] \to n[P \mid Q] \mid m[R]}$$

$$\text{RedOpen}\,\frac{}{\mathsf{open}\ n.P \mid n[Q] \to P \mid Q}$$

$$\text{RedAmb}\,\frac{P \to Q}{n[P] \to n[Q]}$$

$$\text{RedRes}\,\frac{P \to Q}{(\nu n)P \to (\nu n)Q}$$

$$\text{RedPar}\,\frac{P \to Q}{P \mid R \to Q \mid R}$$

$$\text{RedFix}\,\frac{P[\mathbf{fix} X P/X] \to Q}{\mathbf{fix} X P \to Q}$$

Ambient Logic (finite part)

Name variables: $x, \; y, \; \ldots \in \mathcal{V}$ (used with quantifiers)

$u, \; v, \; \ldots \in \mathcal{N} \cup \mathcal{V}$

Propositions:

$$
\begin{array}{lllllllllllll}
A, \; B ::= & \mathsf{T} & | & \neg A & | & A \wedge B & | & \forall x.A & | & \boxdot A & | & & \\
& \mathbf{0} & | & A \mid B & | & A \triangleright B & | & u[A] & | & u \circledR A & | & \text{И} x.A & | \\
& \ldots & & & & & & & & & & &
\end{array}
$$

Semantics

A (closed) proposition is interpreted as a set of processes.

$\boxdot A$ is the standard next time modality:

$$[\![\boxdot A]\!] = \{\, P \mid P \to Q \text{ for some } Q \in [\![A]\!] \,\}$$

$\mathbf{0}$ is satisfied by processes structurally congruent to $\mathbf{0}$:

$$[\![\mathbf{0}]\!] = \{\, P \mid P \equiv \mathbf{0} \,\}$$

A process satisfies $n[A]$ if it is located at n:

$$[\![n[A]]\!] = \{\, P \mid P \equiv n[Q] \text{ and } Q \in [\![A]\!] \,\}$$

$A \mid B$ is satisfied by any process composed of two processes, one satisfying A and the other satisfying B:

$$[\![A \mid B]\!] = \{ P \mid P \equiv Q \mid R,\ Q \in [\![A]\!] \text{ and } R \in [\![B]\!] \}$$

$\triangleright$ (guarantee) is the adjunct of $|$:

$$[\![A \triangleright B]\!] = \{ P \mid \text{for all } Q,\ Q \in [\![A]\!] \Rightarrow Q \mid P \in [\![B]\!] \}$$

$n \circledR A$ is used to reveal a private name:

$$[\![n \circledR A]\!] = \{ P \mid P \equiv (\nu n)Q \text{ and } Q \in [\![A]\!] \}$$

$\text{И}x.A$ is *fresh name quantification*:

$$[\![\text{И}x.A]\!] = \{ P \mid \text{for some } n \notin \mathit{fn}(P, A),\ P \in [\![A[n/x]]\!] \}$$

It's important to know how to compute $\mathit{fn}(A)$.

Adding fixpoints to the logic

In propositional μ-calculus, fixpoint formulas are formed using *propositional variables*. For instance,

$$\nu X.\phi \wedge \boxdot X$$

asserts that the proposition ϕ is always satisfied.

However, for the ambient logic, which is first-order, this is inadequate. Consider

$$\nu Y.n[\mathsf{T}] \mid \text{И} x.Y$$

Any process in the interpretation of $\text{И} x.Y$ should not contain n free, because Y stands for the entire formular thus has n free. But this can not be seen from the syntax.

Solution: use predicate variables

Predicates: functions from names to propositions.

Predicate variables: X, Y, Z, ..., each having an arity.

Extend the logic with the syntactical category *predicates*:

$$F ::= X \quad | \quad (\overline{x})A \quad | \quad \nu X.F \qquad \text{(greatest fixpoint)}$$

Also add a clause to propositions:

$$A ::= \quad \cdots \quad | \quad F(\overline{u}) \qquad \text{(application)}$$

Only use name-closed abstractions:

For any *abstraction* $(\overline{x})A$ it is required that

$$fn(A) = \emptyset \text{ and } fnv(A) \subseteq \{\overline{x}\}$$

Hence $fn(F(\overline{u})) = fn(\overline{u})$.

In general, a fixpoint formula has the form: $(\nu X.(\overline{x})A)(\overline{n})$
($\overline{x}$: formal parameters $\qquad\qquad \overline{u}$: actual parameters)

Now the denotation of formulas are defined with respect to *predicate valuations* $v, v' \ldots$

For each predicate variable X of arity k, $v(X) : \mathcal{N}^k \to 2^{\mathcal{P}}$

Interpretation of predicates:

$$\begin{array}{rcl} [\![X]\!]_v & = & v(X) \\ [\![(\overline{x})A]\!]_v & = & \lambda\overline{n}.[\![A[\overline{n}/\overline{x}]]\!]_v \\ [\![\nu X.F]\!]_v & = & \bigsqcup\{ f \mid f \sqsubseteq [\![F]\!]_{v[X \mapsto f]} \} \end{array}$$

where $\bigsqcup$ and $\sqsubseteq$ are point-wise extension of $\cup$ and $\subseteq$ to $\mathcal{N}^k \to 2^{\mathcal{P}}$. Note that $(\mathcal{N}^k \to 2^{\mathcal{P}}, \sqsubseteq)$ is a complete lattice, for each k.

For application

$$[\![F(\overline{n})]\!]_v = [\![F]\!]_v(\overline{n})$$

An important property of the semantics is monotonicity:

Suppose X is positive in A and F. Then $f \sqsubseteq g$ implies

$$[\![A]\!]_{v[X \mapsto f]} \subseteq [\![A]\!]_{v[X \mapsto g]}$$

$$[\![F]\!]_{v[X \mapsto f]} \sqsubseteq [\![F]\!]_{v[X \mapsto g]}$$

Monotonicity guarantees the existence of fixpoints, and $[\![\nu X.F]\!]_v$ is the greatest one (by Tarski theorem).

Definable operators

Abbreviate $(\nu X.(\overline{x})A)(\overline{n})$ to $\nu X.A[\overline{n}/\overline{x}]$
(X does not occur within the scope of any name quantifier).

The "anytime" modality $\Box$ can be defined from $\boxdot$ and the greatest fixpoint:

$$\Box A \stackrel{def}{=} \nu X.A \land \boxdot X$$

The duals of $\boxdot$ and $\Box$ are $\Diamonddot$ and $\Diamond$ (*sometime*), defined respectively:

$$\Diamonddot A \stackrel{def}{=} \neg\boxdot\neg A \qquad \Diamond A \stackrel{def}{=} \neg\Box\neg A$$

Least fixpoint can be defined as the dual of the greatest fixpoint:

$$\mu X.F \stackrel{def}{=} \neg \nu X.\neg F[\neg X/X]$$

The "somewhere one level downward in space" modality and its dual "anywhere one level downward in space" modality are:

$$\dot{\Diamond} A \stackrel{def}{=} \exists x.x[A] \mid \mathsf{T} \qquad \dot{\Box} A \stackrel{def}{=} \neg \dot{\Diamond} \neg A$$

Now the "somewhere" and "anywhere" modalities $\Diamond$ and $\Box$ can be defined:

$$\Diamond A \stackrel{def}{=} \mu X.A \vee \dot{\Diamond} X \qquad \Box A \stackrel{def}{=} \neg \Diamond \neg A$$

where X does not occur free in A.

An interesting property of the semantics: *fresh names*.

Recall that $P \models \text{И}x.A$ iff $P \models A[n/x]$ for *some* $n \notin \mathit{fn}(P, A)$.

Since n is not free in either P or A, this particular choice of n should not matter: any other name m with $m \notin \mathit{fn}(P, A)$ should equally do.

Proposition 1 (Gabbay-Pitts) $P \models \text{И}x.A$ iff $P \models A[n/x]$ for *every* $n \notin \mathit{fn}(P, A)$.

— "ONE is as good as ALL"

Finite characterisation of universal quantification:

Proposition 2 Suppose $m \notin \mathit{fn}(P, A)$. Then $P \models \forall x.A$ iff $P \models A[n/x]$ for every $n \in \mathit{fn}(A) \cup \{m\}$.

This will be exploited in the model checking algorithm.

Model Checking

Given a process P and a proposition A, determine if $P \models A$.

For the finite fragment of the logic, it has been shown that model checking is

- decidable without composition adjunct ($\triangleright$)

- undecidable with composition adjunct

We shall show that model checking remains decidable for the $\triangleright$-free fragment with fixpoints over "finite-control" processes.

The algorithm

$\mathtt{check}(P, A) =$ case A of

$B \mid C \Rightarrow \bigvee_{P_1 | P_2 \equiv P} \mathtt{check}(P_1, B) \wedge \mathtt{check}(P_2, C)$

$n[B] \Rightarrow$ if $P \equiv n[Q]$ then $\mathtt{check}(Q, B)$ else false

$\boxdot B \Rightarrow \bigwedge_{P \to Q} \mathtt{check}(Q, B)$

$\text{И} x.B \Rightarrow \mathtt{check}(P, B[\mathtt{new}(\mathit{fn}(P, A))/x])$

$\forall x.B \Rightarrow \bigwedge_{n \in \mathit{fn}(P,A) \cup \{\mathtt{new}(\mathit{fn}(P,A))\}} \mathtt{check}(P, B[n/x])$

...

$(\nu X F)(\overline{n}) \Rightarrow$?

When both the processes and logic admit recursion, the main difficulty is to determine when to stop.

We adapt the local model checking algorithm for the propositional μ-calculus by Winskel (1991), generalising the syntax of fixpoint predicate to $\nu X.[I]F$, to include a finite function I from names to sets of processes, with the semantics

$$[\![\nu X.[I]F]\!]_v = \sqcup\{\, f \mid f \sqsubseteq I \sqcup [\![F]\!]_{v[X\mapsto f]}\,\}$$

Intuitively I records the states encountered so far while checking $\nu X.F$.

As a special case, $[\![\nu X.[\emptyset]F]\!]_v = [\![\nu X.F]\!]_v$.

The key property of this semantics:

Lemma If $P \in I(\overline{n})$ (i.e. $(\overline{n}, P) \in I$) then $P \in [\![\nu X.[I]F]\!]_v(\overline{n})$; Otherwise $P \in [\![\nu X.[I]F]\!]_v(\overline{n})$ *iff* $P \in [\![F[\nu X.[\{(\overline{n}, \{P\})\} \sqcup I]F/X]]\!]_v(\overline{n})$.

Now the algorithm:

$\texttt{check}(P, A) =$ case A of

. . .

$(\nu X[I]F)(\overline{n}) \Rightarrow$ if $P \in I(\overline{n})$ then true
else $\texttt{check}(P, F[\nu X[\{(\overline{n}, \{P\})\} \bigsqcup I]F/X](\overline{n}))$

Partial Correctness

Theorem For any process P and closed formula A not containing $\triangleright$, when $\texttt{check}(P, A)$ terminates it returns *true iff* $P \models A$.

Proof

- Most cases follow closely the interpretation of formulas.

- The cases of И and $\forall$ are justified by Propositions 1 and 2, respectively.

- The case of $(\nu X[I]F)(\overline{n})$ is justified by the previous lemma.

Termination (for finite control ambients)

Define $C \prec B$:

- either C is a proper subformular of B
- or $C \equiv F[\nu X[\{(\overline{n}, \{Q\})\} \bigsqcup I]F/X](\overline{n})$ and $B \equiv (\nu X[I]F)(\overline{n})$ with $Q \notin I(\overline{n})$

$\prec$ is well-founded because the number of processes reachable from P is finite.

Each call of `check` leads to a smaller formular w.r.t $\prec$ — guaranteeing termination

Summary

- A spatial logic with both fresh name quantification and fixpoints, based on predicates;
- A simpler and cleaner semantics (no need of “supports/property sets”)
- A decidability result for model checking finite-control ambients against the guarantee-free logic (with fixpoints).

Can be easily adapted for synchronous/asynchronous π-calculus.