Contents

A Semantic Approach to Refinement: Mobile TLA

UML for mobility

- semi-formal graphical notation
- semantics and formal fondation non-obvious
- no notion for reasoning on mobile systems
- no abstract notion of refinement

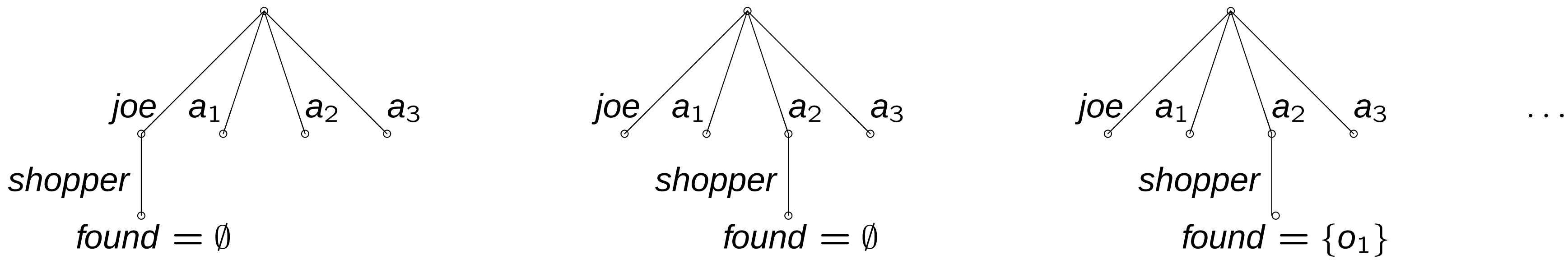
Existing formalisms for mobile systems

- mostly calculi, some with associated logics
- “intensional” semantics, reflecting process structure
- no good notions of refinement

Reactive systems

- transition system semantics (next-state relation + fairness)
- temporal logic properties
- refinement : stuttering invariance

Computational model



Configurations (t, λ)

t finite tree, edges labelled by unique names

λ assigns local states to nodes

Computations $\sigma = (t_0, \lambda_0), (t_1, \lambda_1), \ldots$

Shopping agent specification (1)

Assume: fixed, finite set *Net* of names, $joe \in Net$, $shopper \notin Net$

Network topology

$$Topology \equiv \Box \bigwedge_{n,m \in Net} n\langle m[\mathbf{false}]\rangle$$

all nodes present at top level

Initial condition

$$Init \equiv \wedge\; joe\langle shopper\langle \mathbf{true}\rangle\rangle$$

shopping agent in domain *joe* . . .

$$\wedge\; shopper[ctl = \text{"idle"}]$$

. . . and in "idle" state

Prepare shopper to shop for item *x*

$$Prepare(x) \equiv \wedge\; shopper\langle \mathbf{true}\rangle \wedge \circ shopper\langle \mathbf{true}\rangle$$

shopping is (and stays) here

$$\wedge\; shopper[ctl = \text{"idle"}]$$

state changes from "idle" . . .

$$\wedge\; \circ shopper[ctl = \text{"shopping"}]$$

. . . to "shopping"

$$\wedge\; \circ shopper[target = x \wedge found = \emptyset]$$

initialize *target* and *found*

Shopping agent specification (2)

Remaining state-changing actions

$$GetOffer \equiv \ldots$$ get an offer and insert into *found*

$$PickOffer \equiv \ldots$$ select among offers in *found*

Move among network nodes

$$Move_{n,m} \equiv \wedge\ n\langle shopper\,\langle \mathbf{true}\rangle\rangle$$ shopping agent is in *n*'s domain

$$\wedge\ shopper\,[ctl = \text{“shopping”}]$$ and is in “shopping” state

$$\wedge\ n.shopper \gg m.shopper$$ *shopper* moves to *m*'s domain, preserving local state

Overall specification (ignoring fairness)

$$\begin{aligned} Shopper \equiv\ & \wedge\ Topology \wedge Init \\ & \wedge\ \Box\Big[joe[(\exists x : Prepare(x)) \vee PickOffer] \ \vee\ \bigvee_{n\in Net} n[GetOffer]\Big]_{vars} \\ & \wedge\ \bigwedge_{n\in Net} \Box\Big[\bigvee_{m\in Net} Move_{n,m}\Big]_{-n.shopper} \end{aligned}$$

Spatial extensions of TLA

Formulas evaluated at run σ and name n $\sigma, n \models F$

Explicit name references $m[F]$

- F holds at location m below . . . provided m exists
- Note : m may be arbitrarily deep in subtree

"Everywhere" operator $\overline{\Box} F$

F holds at all nodes of the subtree

Structural modification of trees $\alpha.n \gg \beta.n$

- subtree at αn before transition equals subtree at βn after transition
- local state at moving subtree preserved

System properties

The shopping agent is always at some net location

$$Shopper \Rightarrow \Box \bigvee_{n \in Net} n.shopper \langle \mathbf{true} \rangle$$

The shopper idles only at its home location

$$Shopper \Rightarrow \Box(shopper.ctl = \text{“idle”} \Rightarrow joe.shopper \langle \mathbf{true} \rangle)$$

Refinement of mobile systems

Operation refinement (Action Refinement)

- decompose high-level operations
- represented by implication (stuttering invariance)

Spatial decomposition (Location Refinement)

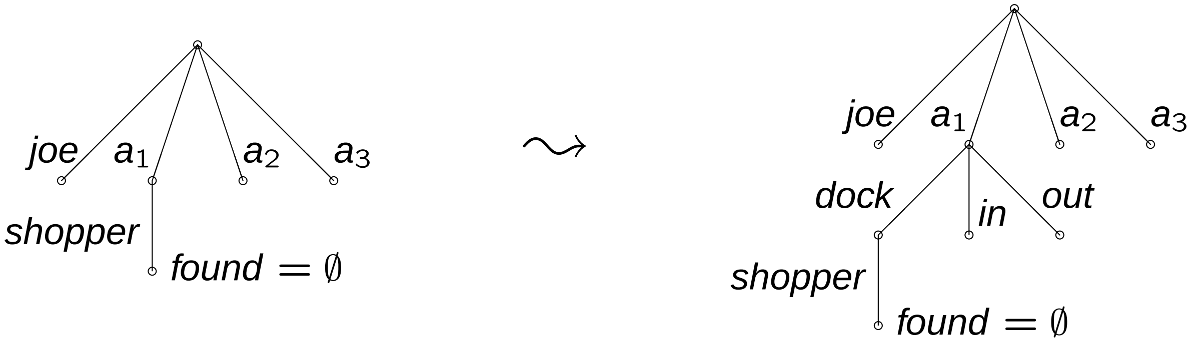
- refine high-level location *n* into a tree (with root named *n*)
- in general also distribute local state of *n*

Virtualisation of locations (Location and Move Refinement)

- implement high-level location *n* by structurally different hierarchy
- preserve external behavior : *n* hidden from high-level interface

Spatial decomposition

Suppose visiting agents are kept in a “dock” location



Still conforms to the original specification

– formula *Shopper* doesn’t mention locations *dock*, *in*, *out*

– location *shopper* is still below location a_1

Spatial decomposition in detail

Refined initial condition

$$DockedInit \equiv \wedge\ joe.dock_{joe}.shopper\,\langle \mathbf{true} \rangle \qquad \text{shopper still in joe's domain}$$
$$\wedge\ shopper\,[ctl = \text{“idle”}] \qquad \text{local state unaffected}$$

shopper still in *joe*'s domain

local state unaffected

Refined move actions

$$SendShopper_n \equiv \wedge\ n.dock_n.shopper\,\langle \mathbf{true} \rangle$$
$$\wedge\ shopper\,[ctl = \text{“shopping”}]$$
$$\wedge\ n.dock_n.shopper \gg n.out_n.shopper$$

stuttering action at high level

$$MoveImpl_{n,m} \equiv \wedge\ n.out_n.shopper\,\langle \mathbf{true} \rangle$$
$$\wedge\ n.out_n.shopper \gg m.in_m.shopper$$

specialization of *Move* action

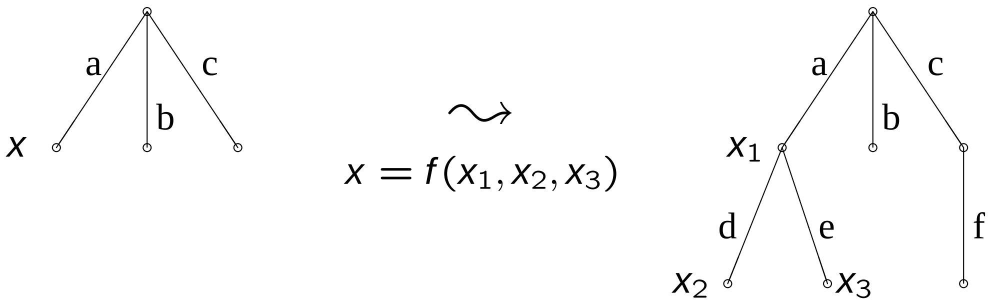
$$RcvShopper_m \equiv \ldots$$

another stuttering transition

The refined specification again implies the original one

Spatial decomposition: general case

Usually, decomposition requires distribution of state

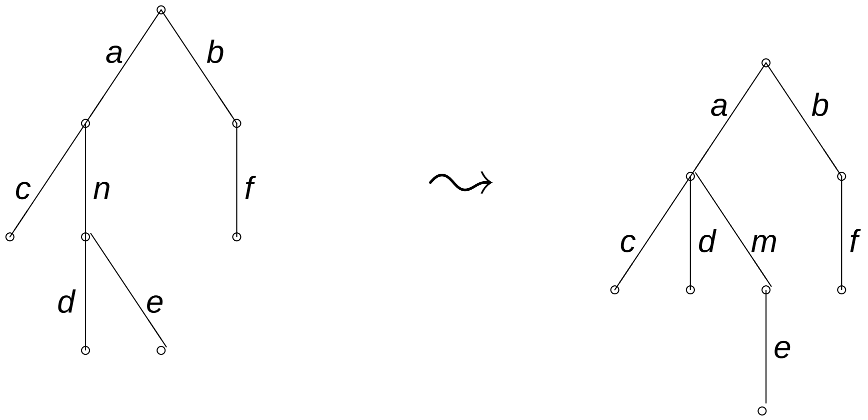


Refinement is then expressed as $\quad Impl \Rightarrow \exists\, a.x : Spec$

local state variable x hidden from high-level interface

Virtualisation of locations

Modify spatial hierarchy



Location *n* hidden from interface $\quad Impl \Rightarrow \exists n : Spec$

preserve external behavior, except for location *n*

SlowShopper : refine move action

Non-atomic moves across network

$$StartMove_n \equiv \wedge\, n.shopper\langle \mathbf{true}\rangle \wedge shopper[ctl = \text{“shopping”}] \wedge n.shopper \gg transit.shopper$$

shopper moves to *transit* $\notin$ *Net*

$$EndMove_m \equiv \wedge\, transit.shopper\langle \mathbf{true}\rangle \wedge transit.shopper \gg m.shopper$$

shopper moves to destination

Implementation does not imply specification

$$\not\models \quad SlowShopper \Rightarrow \Box \bigvee_{n \in Net} n.shopper\langle \mathbf{true}\rangle$$

Solution : hide *shopper* in original specification

$$\models \quad SlowShopper \Rightarrow \exists\, shopper : Shopper$$

Summary and Future Work

Summary

- Simple refinement calculi for activity and sequence diagrams for mobility
- MTLA as a formal basis for a UML notion of refinement: Refinement is implication!

Current Work

- Refinement of other UML diagrams
- Connecting MTLA with UML