Coarse-Grained Interleaving Semantics for SCIMP1

Commands

\[ C_1 \parallel C_2 \text{ means that the actions of } C_1 \text{ and done at the same time as, } \]

\[ \text{in parallel with, those of } C_2 \]

\[ \text{True parallelism hard to model; must handle collisions on resources} \]

\[ \text{What is the meaning of } x := 1 | x := 0 \]

\[ \text{True parallelism exists in real world, so important to model correctly} \]

Interleaving Semantics

Weaker alternative: interleaving semantics

- Each process gets a turn to commit some atomic steps; no preset order of turns, no preset number of actions
- No collision for \( x := 1 | x := 0 \)
  - Yields only \( x\rightarrow 1 \) and \( x\rightarrow 0 \); no collision
- No simultaneous substitution: \( x := y | y := x \) results in \( x \) and \( y \) having the same value; not in swapping their values.

Labeled Transition System (LTS)

A labeled transition system (LTS) is a 4-tuple \( (Q, \Sigma, \delta, I) \) where

- \( Q \) set of states
  - \( Q \) finite or countably infinite
- \( \Sigma \) set of labels (aka actions)
  - \( \Sigma \) finite or countably infinite
- \( \delta \subseteq Q \times \Sigma \times Q \) transition relation
- \( I \subseteq Q \) initial states

Note: Write \( q \xrightarrow{\alpha} q' \) for \( (q, \alpha, q') \in \delta \).
Example: Candy Machine

- \( Q = \{ \text{Start}, \text{Select}, \text{GetMarsBar}, \text{GetKitKatBar} \} \)
- \( I = \{ \text{Start} \} \)
- \( \Sigma = \{ \text{Pay}, \text{ChooseMarsBar}, \text{ChooseKitKatBar}, \text{TakeCandy} \} \)

- \( (\text{Start}, \text{Pay}, \text{Select}) \)
- \( (\text{Select}, \text{ChooseMarsBar}, \text{GetMarsBar}) \)
- \( (\text{GetMarsBar}, \text{TakeCandy}, \text{Start}) \)

Labeled Transition Systems vs Finite State Automata

- LTS have no accepting states
- Every FSA an LTS - just forget the accepting states
- Set of states and actions may be countably infinite
- May have infinite branching

Predecessors, Successors and Determinism

Let \((Q, \Sigma, \delta, I)\) be a labeled transition system.

\[ \text{ln}(q, \alpha) = \{ q' | q' \xrightarrow{\alpha} q \} \]
\[ \text{ln}(q) = \bigcup_{\alpha \in \Sigma} \text{ln}(q, \alpha) \]
\[ \text{Out}(q, \alpha) = \{ q' | q \xrightarrow{\alpha} q' \} \]
\[ \text{Out}(q) = \bigcup_{\alpha \in \Sigma} \text{Out}(q, \alpha) \]

A labeled transition system \((Q, \Sigma, \delta, I)\) is deterministic if

\( |I| \leq 1 \) and \( |\text{Out}(q, \alpha)| \leq 1 \)

Executions, Traces, and Runs

- A partial execution in an LTS is a finite or infinite alternating sequence of states and actions \( \rho = q_0q_1q_2\ldots \) such that
  - \( q_0 \in I \)
  - \( q_{i-1} \xrightarrow{\alpha_i} q_i \) for all \( i \) with \( q_i \) in sequence
- An execution is a maximal partial execution
- A finite or infinite sequence of actions \( \alpha_1\ldots\alpha_n \ldots \) is a trace if there exist states \( q_0\ldots q_n\ldots \) such that the sequence \( q_0\alpha_1q_1\ldots\alpha_nq_n\ldots \) is a partial execution.
- Let \( \rho = q_0\alpha_1q_1\ldots\alpha_nq_n\ldots \) be a partial execution. Then
  \[ \text{trace}(\rho) = \alpha_1\ldots\alpha_n \]
- A finite or infinite sequence of states \( q_0\ldots q_n\ldots \) is a run if there exist actions \( \alpha_1\ldots\alpha_n\ldots \) such that the sequence \( q_0\alpha_1q_1\ldots\alpha_nq_n\ldots \) is a partial execution.
- Let \( \rho = q_0\alpha_1q_1\ldots\alpha_nq_n\ldots \) be a partial execution. Then
  \[ \text{run}(\rho) = q_0\ldots q_n\ldots \]

Example: Candy Machine

- Partial execution:
  \( \rho = \text{Start} \cdot \text{Pay} \cdot \text{Select} \cdot \text{ChooseMarsBar} \cdot \text{GetMarsBar} \cdot \text{TakeCandy} \cdot \text{Start} \)
- Trace: \( \text{trace}(\rho) = \text{Pay} \cdot \text{ChooseMarsBar} \cdot \text{TakeCandy} \)
- Run: \( \text{run}(\rho) = \text{Start} \cdot \text{Select} \cdot \text{GetMarsBar} \cdot \text{Start} \)
Program Transition System

A Program Transition System is a triple \((S, T, \text{init})\)
- \(S = (G, D, F, \phi, R, p)\) is a first-order structure over signature \(G = (V, F, \alpha, R, \alpha_r)\)
- \(cS\) used to interpret expressions and conditionals
- \(T\) is a finite set of conditional transitions of the form
  \[ g \rightarrow (v_1, \ldots, v_n) = (e_1, \ldots, e_n) \]
  where \(v_i \in V\) distinct, and \(e_i\) term in \(G\), for \(i = 1 \ldots n\)
- \(\text{init}\) initial condition asserted to be true at start of program

Example: Traffic Lights

\[ V = \{\text{Turn, NSC, EWC}\}, F = \{\text{NS, EW, Red, Yellow, Green}\} \text{ (all arity 0),} \]
\[ R = \{\} \]

\[ \begin{align*}
\text{NSG} & \quad \text{Turn} = \text{NS} \land \text{NSC} = \text{Red} \rightarrow \text{NSC} = \text{Green} \\
\text{NSY} & \quad \text{Turn} = \text{NS} \land \text{NSC} = \text{Green} \rightarrow \text{NSC} = \text{Yellow} \\
\text{NSR} & \quad \text{Turn} = \text{NS} \land \text{NSC} = \text{Yellow} \rightarrow (\text{Turn, NSC}) = (\text{EW, Red}) \\
\text{EWR} & \quad \text{Turn} = \text{EW} \land \text{EWC} = \text{Red} \rightarrow \text{EWC} = \text{Green} \\
\text{EWY} & \quad \text{Turn} = \text{EW} \land \text{EWC} = \text{Green} \rightarrow \text{EWC} = \text{Yellow} \\
\end{align*} \]

\[ \text{init} = (\text{NSC} = \text{Red} \land \text{EWC} = \text{Red} \land (\text{Turn} = \text{NS} \lor \text{Turn} = \text{EW})) \]

Mutual Exclusion (Attempt)

\[ \begin{align*}
P1 & : m1 : \text{while true do} \\
& \quad m2 : \text{p1}(!\text{not in crit sect}) \\
& \quad m3 : c1 := 0 \\
& \quad m4 : \text{wait}(c2 = 1) \\
& \quad m5 : r1(!\text{in crit sect}) \\
& \quad m6 : c1 := 1 \\
& \quad m7 : \text{ od} \\
\end{align*} \]

\[ \begin{align*}
P2 & : n1 : \text{while true do} \\
& \quad n2 : \text{p2}(!\text{not in crit sect}) \\
& \quad n3 : c2 := 0 \\
& \quad n4 : \text{wait}(c1 = 1) \\
& \quad n5 : r2(!\text{in crit sect}) \\
& \quad n6 : c2 := 1 \\
& \quad n7 : \text{ od} \\
\end{align*} \]

Interpreting PTS as LTS

Let \((S, T, \text{init})\) be a program transition system. Assume \(V\) finite, \(D\) at most countable.
- Let \(Q = V \rightarrow D\), interpreted as all assignments of values to variables
  - Can restrict to mappings \(q\) where \(v\) and \(q(v)\) have same type
- Let \(\Sigma = T\)
- Let \(\delta = \{(q, g : (v_1, \ldots, v_n) : (e_1, \ldots, e_n), q') \mid \}
  - \(M_q(g)\) \&
  - \((\forall i \leq n, q'(v_i) = T_q(e_i))\) \&
  - \((\forall v \notin \{v_1, \ldots, v_n\}, q'(v) = q(v))\)\)
- \(I = \{qT_q(\text{init}) = T\}\)

Example: Traffic Lights
Examples (cont)

- LTS for traffic light has $3 \times 3 \times 2 = 18$ possible well typed states:
  - Is it possible to reach a state where NSC $\neq$ Red $\land$ EWC $\neq$ Red from an initial state?
  - If so, what sequence of actions allows this?
  - Do all the immediate predecessors of a state where NSC $=$ Green $\lor$ EWC $=$ Green satisfy NSC $= \text{Red} \land$ EWC $= \text{Red}$?
  - If not, are any of those offending states reachable from and initial state, and if so, how?
- LTS for Mutual Exclusion has $6 \times 6 \times 2 = 144$ possible well-typed states.
  - Is it possible to reach a state where $pc1 = m5 \land pc2 = n5$?
  - How can we state these questions rigorously, formally?
  - Can we find an algorithm to answer these types of questions?

Linear Temporal Logic - Syntax

$\phi ::= p | (\phi) | \neg \phi | \phi \land \phi' | \phi \lor \phi' | \Diamond \phi | \Box \phi | \phi U \psi | \phi V \psi$

- $p$ – a proposition over state variables
- $\Diamond$ – “next”
- $\Box$ – “always”, “forever”
- $\bowtie$ – “box”, “eventually”, “sometime”

Formal LTL Semantics

Given:
- $G = (V, F, af, R, ar)$ signature expressing state propositions
- $Q$ set of states,
- $M$ modeling function over $Q$ and $G$: $M(q, p)$ is true iff $q$ models $p$.

LTS Semantics: The Idea

- $\sigma \models p$ iff $pc1 = m5 \land pc2 = n5$
- $\sigma \models \neg p$ iff $pc1 = m5 \land pc2 = n5$
- $\sigma \models \phi U \psi$ iff for some $k$, $\sigma_k \models \psi$ and for all $i < k$, $\sigma_i \models \phi$
- $\sigma \models \phi V \psi$ iff for some $k$, $\sigma_k \models \psi$ and for all $i \leq k$, $\sigma_i \models \phi$.
- $\sigma \models \Diamond \phi$ if for all $i$, $\sigma_i \models \phi$
- $\sigma \models \Box \phi$ if for some $i$, $\sigma_i \models \phi$
- $\sigma \models \Box \phi$ if for all $i$, $\sigma_i \models \phi$
- $\sigma \models \phi V \psi$ if $\sigma \models \phi$ or $\sigma \models \psi$

Examples (cont)

- $\text{LTS for traffic light has } 3 \times 3 \times 2 = 18 \text{ possible well-typed states}$
- $\text{Is it possible to reach a state where } pc1 = m5 \land pc2 = n5$?
- $\text{If so, how can we state these questions rigorously, formally?}$
- $\text{Can we find an algorithm to answer these types of questions?}$

Some Common Combinations

- $\Box \Diamond p$ “$p$ will hold infinitely often”
- $\Box \Box p$ “$p$ will continuously hold from some point on”
- $(\Diamond \Box p) \Rightarrow (\Box \Box q)$ “if $p$ happens infinitely often, then so does $q$”
### Some Equivalences

- $\square (\varphi \land \psi) = (\square \varphi) \land (\square \psi)$
- $\Diamond (\varphi \lor \psi) = (\Diamond \varphi) \lor (\Diamond \psi)$
- $\square \varphi = F \lor \lnot \varphi$
- $\Diamond \varphi = T \lor \lnot \varphi$
- $\varphi \lor \psi = \lnot \lnot (\varphi \lor \psi)$
- $\varphi \land \psi = \lnot \lnot (\varphi \land \psi)$
- $\lnot \Diamond \varphi = \square (\lnot \varphi)$
- $\lnot \square \varphi = \Diamond (\lnot \varphi)$

### Some More Equivalences

- $\square \varphi = \varphi \land (\square \varphi)$
- $\Diamond \varphi = \varphi \lor (\Diamond \varphi)$
- $\varphi \lor \psi = (\varphi \land (\varphi \lor \psi))$
- $\varphi \land \psi = (\varphi \land (\varphi \land \psi))$
- $\Diamond \varphi = \varphi \lor (\Diamond \varphi)$
- $\Square \Diamond \varphi = \Diamond (\Diamond \varphi)$
- $\Diamond \Square \varphi = \Square (\Diamond \varphi)$
- $\Square \Diamond \varphi = \Diamond (\Square \varphi)$

### Traffic Light Example

**Basic Behavior:**
- $\square ((\text{NSC} = \text{Red}) \lor (\text{NSC} = \text{Green}) \lor (\text{NSC} = \text{Yellow}))$
- $\square ((\text{NSC} = \text{Red}) \Rightarrow ((\text{NSC} \neq \text{Green}) \land (\text{NSC} \neq \text{Yellow})))$
- Similarly for Green and Red
- $\square ((\text{NSC} = \text{Red}) \land (\square \text{NSC} \neq \text{Red})) \Rightarrow \square (\text{NSC} = \text{Green}))$
- Same as $\square ((\text{NSC} = \text{Red}) \Rightarrow ((\text{NSC} = \text{Red}) \lor (\text{NSC} = \text{Green})))$
- $\square ((\text{NSC} = \text{Green}) \land (\square \text{NSC} \neq \text{Green})) \Rightarrow \square (\text{NSC} = \text{Yellow}))$
- $\square ((\text{NSC} = \text{Yellow}) \land (\square \text{NSC} \neq \text{Yellow})) \Rightarrow \square (\text{NSC} = \text{Red}))$
- Same for EWC

### Proof System for LTL

First step: View $\varphi \lor \psi$ as macro: $\varphi \lor \psi = \lnot (\lnot \varphi) \lor (\lnot \psi)$

Second Step: Extend all rules of Prop Logic to LTL

Third Step: Add one more rule: $\Diamond \varphi \land \varphi$

Fourth Step: Add a collection of axioms (a sufficient set of 8 exists)

Result: a sound and relatively complete proof system