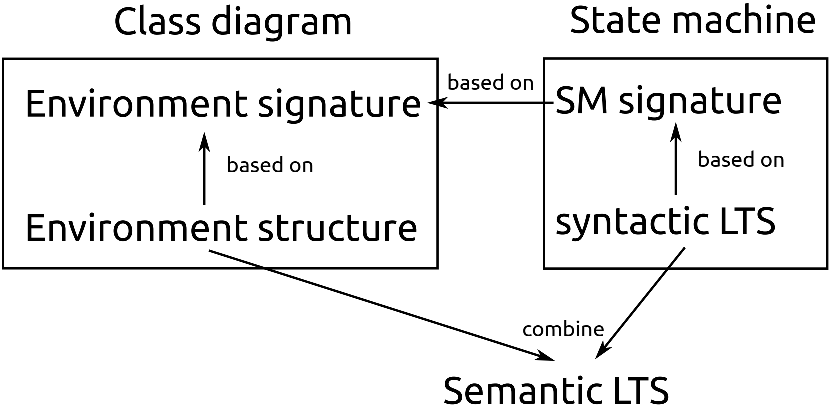
Putting it together, semantically: Semantics of UML state machines

Till Mossakowski[1]

Otto-von-Guericke Universität Magdeburg, Germany

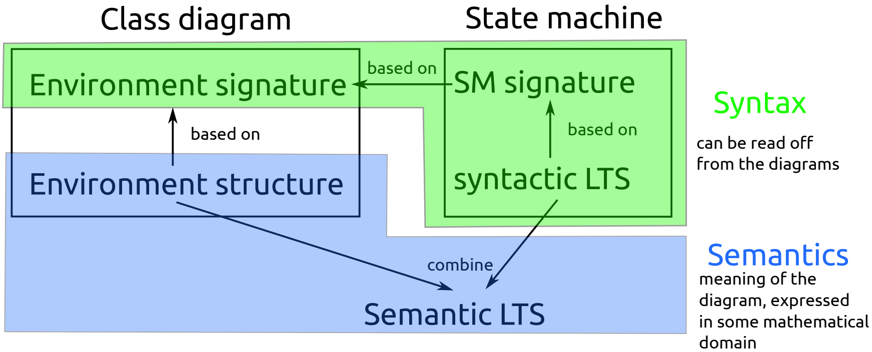
June 28, 2016

Overview



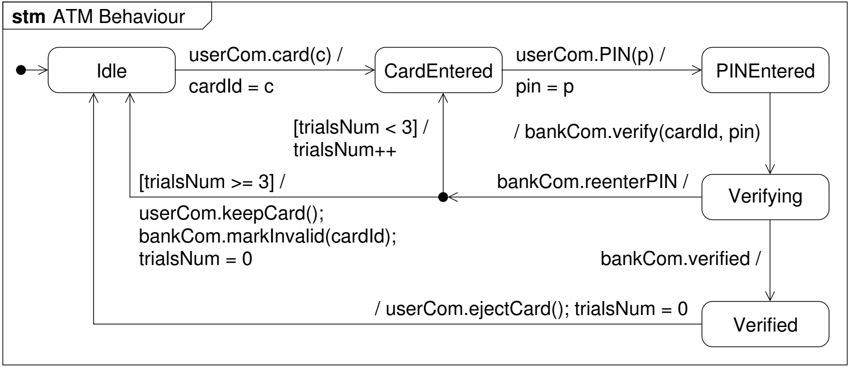
LTS = labeled transition system

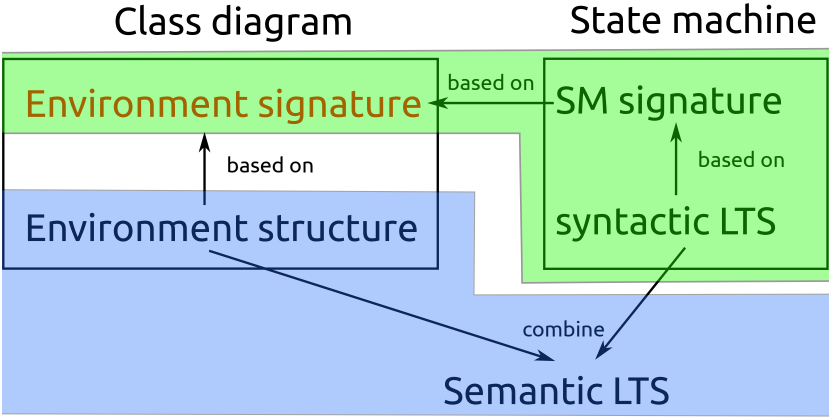
Overview — Syntax & Semantics



LTS = labeled transition system

A Sample State Machine





LTS = labeled transition system

Environment Signatures

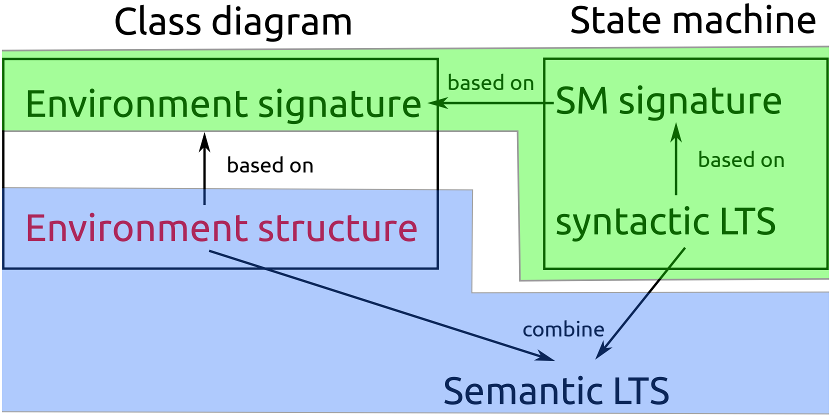
An environment signature is a triple of sets

$$H = (G_H, A_H, M_H)$$

of guards, actions, and messages.

Guards: formulas in some logical language, e.g. OCL.
Actions (effects): operations of class diagram, assignments of attributes etc.
Messages (triggers): signals and operations of class diagram

Overview



LTS = labeled transition system

Environment Structures

Given a signature $H = (G_H, A_H, M_H)$,
an environment structure Ω is given by:

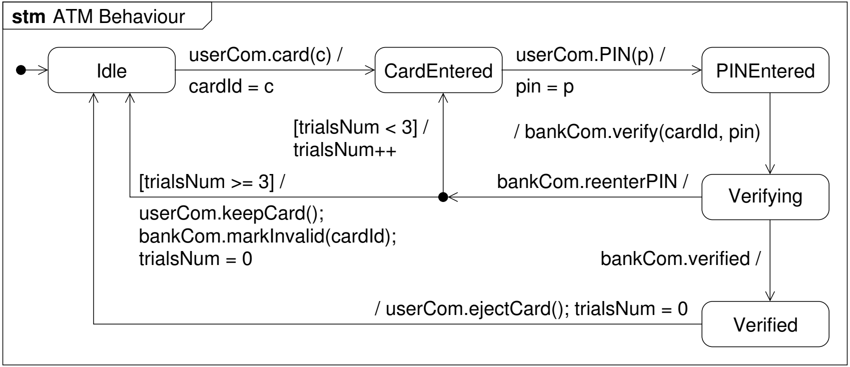
$$\Omega = (|\Omega|, \models_\Omega \subseteq |\Omega| \times G_H, \alpha_\Omega \subseteq |\Omega| \times A_H \times \wp(M_H) \times |\Omega|) \ ,$$

where

- $|\Omega|$: set of data states,
- $\omega \models_\Omega g$: state $\omega \in |\Omega|$ satisfies guard g,
- $(\omega, a, \overline{m}, \omega') \in \alpha_\Omega$, also written $\omega \xrightarrow[\Omega]{a, \overline{m}} \omega'$: action a leads from state $\omega \in |\Omega|$ to state $\omega' \in |\Omega|$ producing the set of messages $\overline{m} \subseteq M_H$.

Example: take $|\Omega|$ to be the data states of a UML class diagram.
Actions a can be e.g. variable updates.

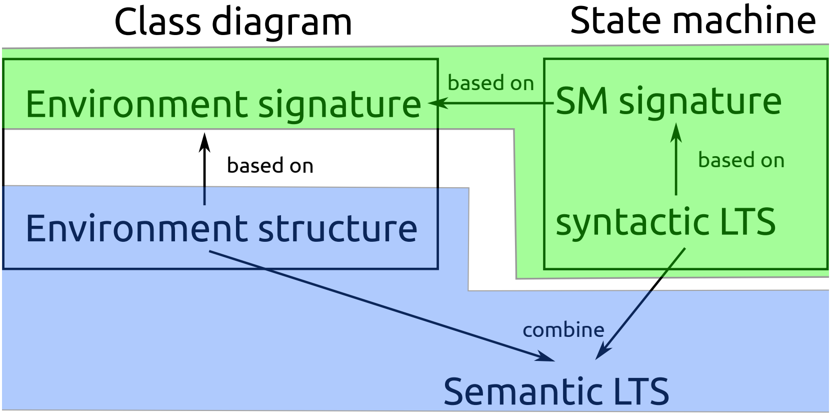
A Sample State Machine



Signature for the Sample State Machine

Environment signature:

guards	true, trialsNum ≤ 3,
actions	user.ejectCard(); trialsNum $= 0$, trialsNum$++$,
messages	user.ejectCard(), bank.markInvalid(cardId)



LTS = labeled transition system

Labeled Transition Systems

Definition (Labeled Transition System)

A labeled transition system LTS is a tuple $(S, L, \rightarrow, I)$, where

- S is a set of states,
- L is a set of actions,
- $\rightarrow \subseteq S \times L \times S$ is a transition relation, and
- $I \subseteq S$ is a set of initial states.

Optionally, there can also be a set of final states (in this case, an LTS is the same a a finite automaton).
We write $s \xrightarrow{a} s'$ for $(s, a, s') \in \rightarrow$.

Definition (Direct successors)

$Post(s, a) = \{s' \in S | s \xrightarrow{a} s'\}$ (for $s \in S$, $a \in L$)

Definition (Deterministic LTS)

LTS is deterministic, if $|I| = 1$ and $|Post(s, a)| \leq 1 \;\; \forall \; s \in S, \; a \in L$

Runs of Labeled Transition Systems

Definition (Finite run)

Given an LTS $(S, L, \to, I)$, a finite run ρ is a finite alternating sequence of states and actions starting with some $s_0 \in I$ and ending with a state

$$\rho = s_0 a_1 s_1 \ldots a_n s_n \text{ such that } s_i \xrightarrow{a_{i+1}} s_{i+1}$$

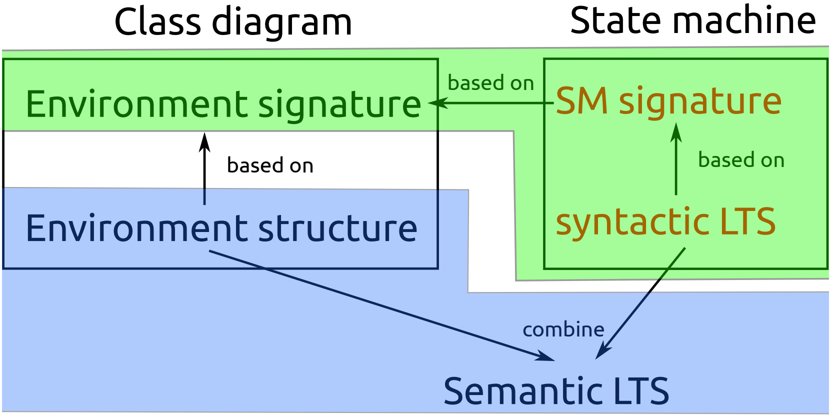
for all $0 \leq i < n$. $n \geq 0$ is the length of the run.

Definition (Infinite run)

Given an LTS $(S, L, \to, I)$, an infinite run ρ is a infinite alternating sequence of states starting with some $s_0 \in I$

$$\rho = s_0 a_1 s_1 a_2 s_2 \ldots \text{ such that } s_i \xrightarrow{a_{i+1}} s_{i+1}$$

for all $0 \leq i$.



LTS = labeled transition system

State Machines as Labeled Transition Systems

Given: $H = (G_H, A_H, M_H)$ environment signature.

A state machine signature is given by a pair of sets: $\Sigma = (E_\Sigma, S_\Sigma)$ (events and states) with $E_\Sigma \cap S_\Sigma = \emptyset$.

Labels: $L = (E_\Sigma \cup S_\Sigma) \times G_H \times A_H$

triggering event (declared or completion event), guard, action

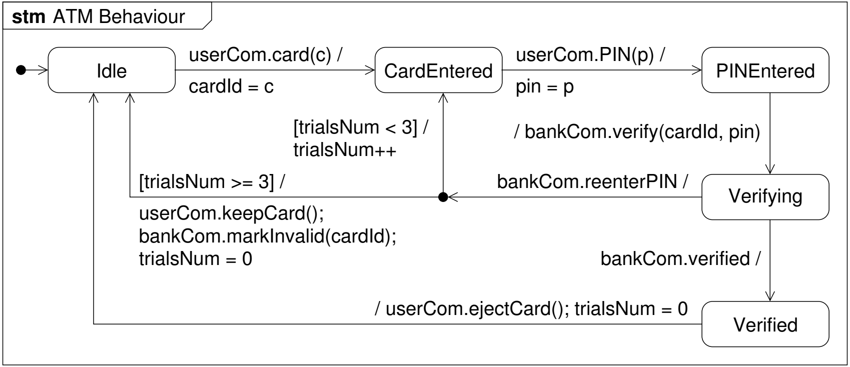
Syntactic labeled transition system of a state machine:

$$(S_\Sigma, L, T \subseteq S_\Sigma \times L \times S_\Sigma, \{s_0\})$$

- T: transition relation, representing transitions from a state to another state.
- s_0: initial state

Note: for simplicity, we omit hierarchical states.

A Sample State Machine



Syntactic LTS for Sample State Machine

Signature: $(E_{\mathsf{ATM}}, S_{\mathsf{ATM}})$ with

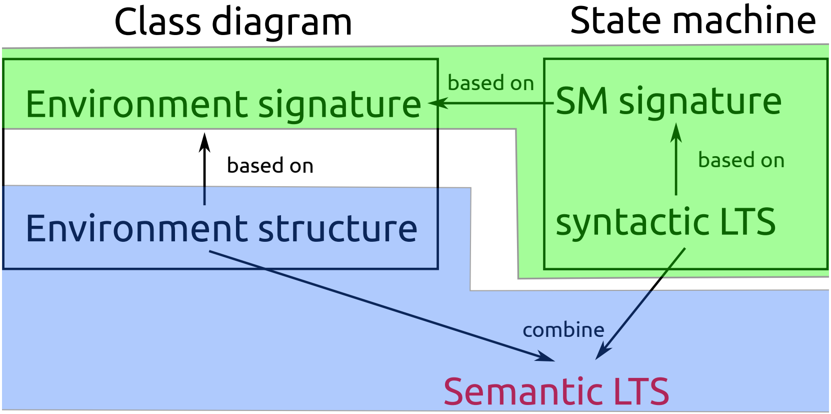
$$E_{\mathsf{ATM}} = \{\mathsf{card}, \mathsf{PIN}, \mathsf{reenterPIN}, \mathsf{PINVerified}\}$$
$$S_{\mathsf{ATM}} = \{\mathsf{Idle}, \mathsf{CardEntered}, \mathsf{PINEntered}, \mathsf{Verifying}, \mathsf{PINVerified}\}$$

The syntactic LTS of the state machine:

$$\begin{aligned}(\{(&\mathsf{Idle}, (\mathsf{card}, \mathsf{true}, \mathsf{cardId} = \mathsf{c}), \mathsf{CardEntered}),\\ &(\mathsf{CardEntered}, (\mathsf{PIN}, \mathsf{true}, \mathsf{pin} = \mathsf{p}), \mathsf{PINEntered}),\\ &(\mathsf{PINEntered}, (\mathsf{PINEntered}, \mathsf{true}, \mathsf{bank.verify(cardId,\ pin)}), \mathsf{Verifying}\\ &(\mathsf{Verifying}, (\mathsf{reenterPIN}, \mathsf{trialsNum} < 2, \mathsf{trialsNum}{+}{+}),\\ &\qquad \mathsf{CardEntered}), \ldots\}, \{\mathsf{Idle}\})\end{aligned}$$

In particular, PINEntered occurs both as a state and as a completion event in the third transition. The junction pseudostate for making the decision whether $\mathsf{trialsNum} < 2$ or $\mathsf{trialsNum} \geq 2$ has been resolved by combining the transitions.

Overview



LTS = labeled transition system

The Induced Semantic Labeled Transition System

Syntactic LTS Θ: control states S_Σ

Semantic LTS Δ_Θ: control and data states:

States: $C = |\Omega| \times \wp(E_\Sigma \cup S_\Sigma) \times S_\Sigma$
environment state, an event pool, and a control state
Labels: $L = \wp(M_H)$ set of messages

The event pool may contain both events declared in the signature (from signals and operations) and completion events (represented by states).

The Induced Semantic Labeled Transition System, cont'd

Transition relation:

$$(\omega, p :: \overline{p}, s) \xrightarrow[\Delta_\Theta]{\overline{m} \backslash E_\Sigma} (\omega', \overline{p} \lhd ((\overline{m} \cap E_\Sigma) \cup \{s'\}), s') \quad \text{if}$$

$$\exists s \xrightarrow[T]{p[g]/a} s' \,.\, \omega \models g \wedge \omega \xrightarrow[\Omega]{a, \overline{m}} \omega'$$

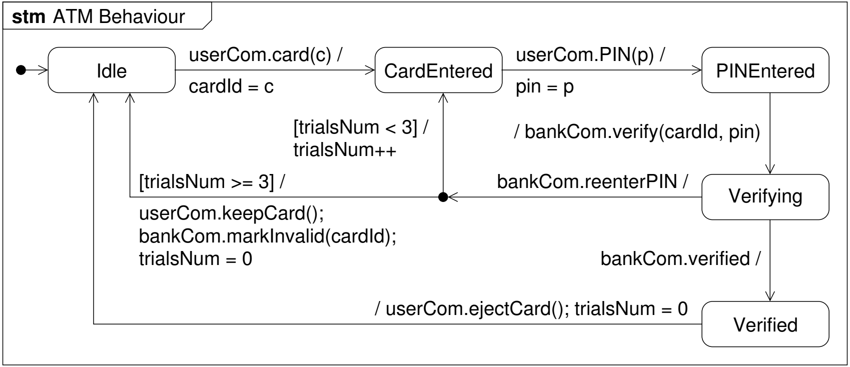
$$(\omega, p :: \overline{p}, s) \xrightarrow[\Delta_\Theta]{\emptyset} (\omega, \overline{p}, s) \quad \text{if}$$

$$\forall s \xrightarrow[T]{p'[g]/a} s' \,.\, p \neq p' \vee \omega \not\models g$$

$p \uplus \overline{p}$: p is next event to be processed
$\overline{p} \lhd \overline{p}'$: adds events $\overline{p}'$ to pool $\overline{p}$
$\overline{m} \cap (M_H \setminus E_\Sigma)$: messages emitted
$(\overline{m} \cap E_\Sigma) \cup \{s'\}$: accepted events in E_Σ and completion event when entering state s' are added to the event pool.

When no transition is triggered by the current event, the event is discarded (this will happen, in particular, to all superfluously generated completion events).

Sample State Machine



Protocol state machines

Protocol state machines: pre- and a postcondition instead of guards and effects.
Events that do not fire a transition are an error.
The syntactic LTS is changed to:

$$(T \subseteq S_\Sigma \times (G_H \times E_\Sigma \times G_H \times \wp(M_H)) \times S_\Sigma, \{s_0\})$$

where

- the two occurrences of G_H represent the pre- and the post-conditions,
- $\wp(M_H)$ represents the messages that have to be sent out in executing the triggering event