An Institutional Approach to Communicating UML State Machines

Tobias Rosenberger

Swansea University

VERIMAG, Université Grenoble Alpes

Alexander Knapp

Universität Augsburg

Markus Roggenbach

Swansea University

Heterogeneous Modelling

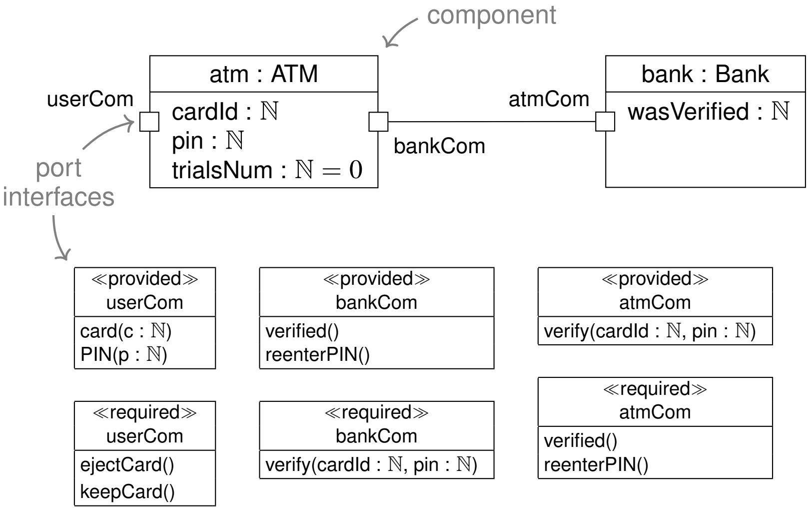
"Unified Modeling Language" (UML)

- 14 different sub-languages (diagram types)
- loosely integrated by a common meta-model (abstract syntax)

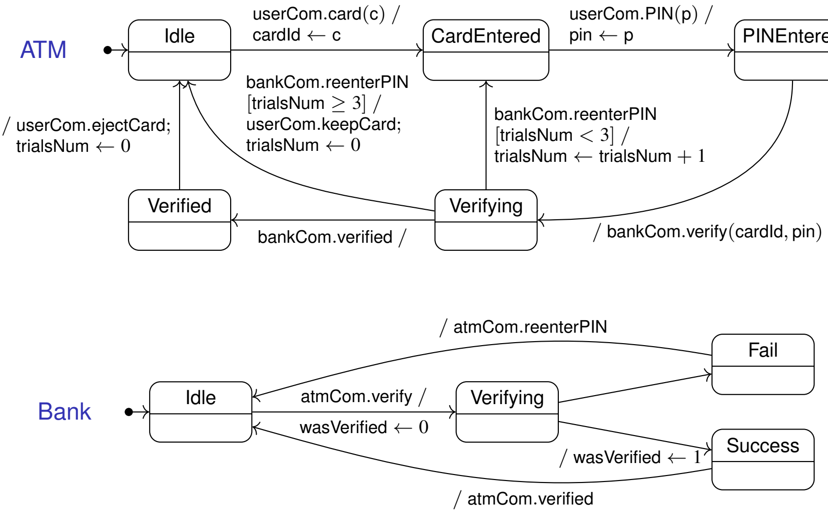
Goal: UML as a heterogeneous modelling language

- separate semantics for different sub-languages of the UML as institutions
  - abstract model-theoretic framework (J. A. Goguen, R. M. Burstall)
- linkage by institution (co-)morphisms
- integration into the "Heterogeneous Tool Set" (HETS)
  - analysis and proof support for multi-logic specifications (SAT solvers, automated and interactive theorem provers, model finders, model checkers)

Communicating UML State Machines (1)



Communicating UML State Machines (2)



Semantics of Communicating State Machines

OMG's "reference" executable semantics based on fUML

- Precise Semantics of UML Composite Structures (PSCS 1.2), 2019
- Precise Semantics of UML State Machines (PSSM 1.0), 2019

Simulation, model checking, and theorem proving

- H. Grönninger, B. Rumpe (Isabelle/HOL, 2010)
- I. Ober, I. Dragomir (OMEGA2, IF Tool Suite, 2011)
- S. Liu, Y. Liu, É. André, C. Choppy, J. Sun, B. Wadhwa, J. S. Dong (USM^2C, 2013)
- F. Mazzanti, A. Ferrari, G. O. Spagnolo (KandISTI/UMC, 2017)

Institutional approaches

- A. K., T. Mossakowski, M. Roggenbach, M. Glauer (simple state machines, 2015)
- A. K., T. Mossakowski (relation with interactions, 2017)

Institutions

Institution $(\mathbb{S}, Str, \mathrm{Sen}, \models)$

- category $\mathbb{S}$ of signatures
- indexed category $Str\colon \mathbb{S}^{\mathrm{op}} \to \mathrm{Cat}$ of structures
- sentences functor $\mathrm{Sen}\colon \mathbb{S} \to \mathrm{Set}$
- family $\models\, = (\models_\Sigma \subseteq |Str(\Sigma)| \times \mathrm{Sen}(\Sigma))_{\Sigma \in |\mathbb{S}|}$ of satisfaction relations

such that satisfaction condition holds for all $\sigma\colon \Sigma \to \Sigma'$ in $\mathbb{S}$, $M' \in Str(\Sigma')$, and $\varphi \in \mathrm{Sen}(\Sigma)$

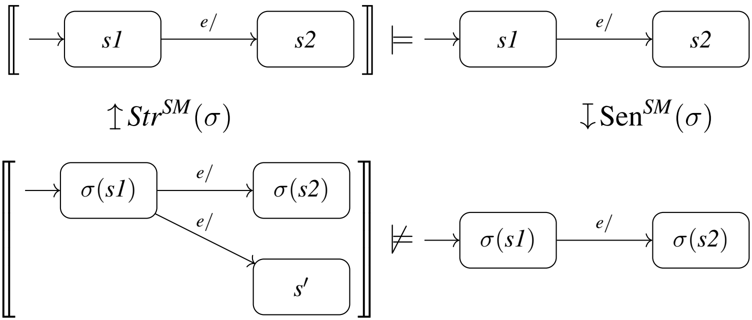
$$Str(\sigma)(M') \models_\Sigma \varphi \iff M' \models_{\Sigma'} \mathrm{Sen}(\sigma)(\varphi)$$

"Truth is invariant under change of notation."

Institutional Semantics for State Machines

State machines both as structures and sentences (A. K. et al., 2015, 2017) requires injective signature morphisms

- and surjective on states



Grand (De-)Tour: $\mathcal{M}^{\downarrow}_{\mathcal{D}}$

Event/data-based hybrid modal logic $\mathcal{M}^{\downarrow}_{\mathcal{D}}$

- data: state and transition predicates
- events: input and output with parameters
- modal: box and diamond for event, data, and control state changes
- hybrid: direct access to control state

(building on R. Hennicker, A. K., A. Madeira, 2019 and extending T. Rosenberger, A. K., M. Roggenbach, 2020)

- $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ forms an institution.
- $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ can be translated into CASL by a theoroidal institution comorphism.
- $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ can fully express UML state machines up to isomorphism.
- $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ can be lifted to composite structures institution $\mathrm{CS}(\mathcal{M}^{\downarrow}_{\mathcal{D}})$.
- theorem proving support for $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ and $\mathrm{CS}(\mathcal{M}^{\downarrow}_{\mathcal{D}})$ via HETS

$\mathcal{M}^{\downarrow}_{\mathcal{D}}$: Data, Events, Signatures, Structures

Data described by fixed CASL-specification Dt with single model $\mathcal{D}$

- state predicates $\mathscr{F}^{\mathcal{D}}_{A,X}$ for attributes A
- transition predicates $\mathscr{F}^{2\mathcal{D}}_{A,X}$ for attributes A and their primed variants

Event/data signature $\Sigma = (I, O, A)$

- input and output event signatures with Dt-sorted variables
  - yield input/output messages $\hat{I}$ and $\hat{O}$ for variable valuations
- data signature A of Dt-sorted attributes

Σ-event/data structure $M = (\Gamma, R, \Gamma_0, \omega)$ transition system

- configurations $\Gamma \subseteq C \times D$ of control and data states
- transition relation $R = (R_{\hat{\imath},\hat{O}} \subseteq \Gamma \times \Gamma)_{\hat{\imath} \in \hat{I}(\Sigma), \hat{O} \in \hat{O}(\Sigma)^*}$
- initial configurations $\Gamma_0 \subseteq \Gamma$ such that all Γ are R-reachable
- data state labelling $\omega \colon D \to \Omega(A(\Sigma))$

$\mathcal{M}^{\downarrow}_{\mathcal{D}}$: Formulæ and Sentences

Σ-event/data formulæ $\mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma,S}$ over state variables S

- φ — data state sentence
- s — control state test
- $\downarrow s \,.\, \varrho$ — control state binding
- $(@^{J,N} s)\varrho$ — "jump" relativised to input J and output N
- $\Box^{J,N} \varrho$ — "globally" relativised to input J and output N
- $\langle i \,/\!\!/\, [O]_N : \psi \rangle \varrho$ — "diamond" for some input i and some output O (with possibly some N) satisfying ψ
- $\langle\!| i : \phi \,/\!\!/\, [O]_N : \psi |\!\rangle \varrho$ — "strong diamond" for all input i satisfying ϕ and some output O (with possibly some N) satisfying ψ
- $\neg\varrho$, $\varrho_1 \vee \varrho_2$

Σ-event/data sentences $\mathrm{Sen}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}(\Sigma)$ closed formulæ

$\mathcal{M}^{\downarrow}_{\mathcal{D}}$: Satisfaction Relation

Satisfaction over Σ-structure $M = (\Gamma, R, \Gamma_0, \omega)$, state variable assignment $v \colon S \to C(M)$, in a configuration γ

- $M, v, \gamma \models^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma,S} \varphi$ iff $\omega(\gamma), \emptyset \models^{\mathcal{D}}_{A(\Sigma),\emptyset} \varphi$
- $M, v, \gamma \models^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma,S} s$ iff $v(s) = c(\gamma)$
- …
- $M, v, \gamma \models^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma,S} \langle\!| i : \phi /\!\!/ [O]_N : \psi |\!\rangle \varrho$ iff for all $\beta \colon X(i) \to \mathcal{D}$ with $\omega(\gamma), \beta \models^{\mathcal{D}}_{A(\Sigma),X(i)} \phi$ there are $\beta' \colon X(O) \to \mathcal{D}$, $\hat{O}' \in O(\beta') \parallel \hat{N}$ with $\hat{N} \in \hat{O}(N)^*$, and $\gamma' \in \Gamma$ such that $(\gamma, \gamma') \in R_{i(\beta),\hat{O}'}$, $(\omega(\gamma), \omega(\gamma')), \beta \cup \beta' \models^{2\mathcal{D}}_{A(\Sigma),X(i)\cup X(O)} \psi$, and $M, v, \gamma' \models^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma,S} \varrho$
- …

Institution $(\mathbb{S}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}, Str^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}, \mathrm{Sen}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}, \models^{\mathcal{M}^{\downarrow}_{\mathcal{D}}})$

- due to relativisations

Theoroidal Institution Comorphism from $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ to CASL

A theoroidal comorphism embeds a "poorer" logic into a "richer" logic.

- signatures translated to a presentation of a signature and sentences

Apply "standard translation" from modal logics to first-order logic

- Signature translation $\nu^{Sig} : \mathbb{S}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}} \to \mathrm{Pres}^{\text{CASL}}$
  - uses a theory presentation for data-based transition systems over input/output-events in the algebraic specification language CASL
  - predicate trans for transitions, init for initial configurations
- Model translation $\nu^{\mathrm{Mod}}_{\Sigma} : \mathrm{Mod}^{\text{CASL}}(\nu^{Sig}(\Sigma)) \to Str^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}(\Sigma)$
  - extracts reachable part of a CASL-model along trans
- Sentence translation $\nu^{\mathrm{Sen}}_{\Sigma} : \mathrm{Sen}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}(\Sigma) \to \mathrm{Sen}^{\text{CASL}}(\nu^{\mathbb{S}}(\Sigma))$
  - uses formula translation $\nu^{\mathscr{F}}_{\Sigma,S,g} : \mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma,S} \to \mathscr{F}^{\text{CASL}}_{\nu^{\mathbb{S}}(\Sigma),S\cup\{g\}}$ that realises "standard translation" for current configuration g
  - requires that evaluation starts in an initial state

Proving $\mathcal{M}^{\downarrow}_{\mathcal{D}}$-invariants in CASL

Full generality of theoroidal comorphism not always needed

Proposition For proving that $\Box inv^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}$ for state formula $inv^{\mathcal{M}^{\downarrow}_{\mathcal{D}}} \in \mathscr{F}^{\mathcal{D}}_{A(\Sigma),\emptyset}$ holds via CASL, it suffices to prove that a generalised invariant inv^{CASL} with

(I0) $\forall g : \mathrm{Conf}\,.\, inv^{\text{CASL}}(g) \Rightarrow \mathscr{F}^{\text{CASL}}_{\nu^{\mathbb{S}}(\Sigma),A(\Sigma)(g)}(inv^{\mathcal{M}^{\downarrow}_{\mathcal{D}}})$

satisfies the following induction scheme:

(I1) $\forall g : \mathrm{Conf}\,.\, \mathrm{init}(g) \Rightarrow inv^{\text{CASL}}(g)$

(I2) $\forall g, g' : \mathrm{Conf}; i \in \mathrm{InEvt}; O \in List[\mathrm{OutEvt}]\,.$
$inv^{\text{CASL}}(g) \wedge \mathrm{trans}(g, i, O, g') \Rightarrow inv^{\text{CASL}}(g')$

Simple UML State Machines (1)

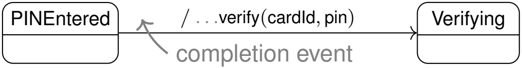
Simple UML state machine $U = (\Sigma, C, T, (c_0, \varphi_0))$ over event/data signature Σ

- finite sets of control states C and transition specifications T of the form $(c, \phi, i(X), o_1(X_1), \ldots, o_m(X_m), \psi, c')$ with
  - source and target control states $c, c' \in C$,
  - input $i(X) \in I(\Sigma)$ and output events $o_1(X_1), \ldots, o_m(X_m) \in O(\Sigma)$ with disjoint variables (i.e., $X \cap \bigcup_{1 \leq k \leq m} X_m = \emptyset$),
  - precondition $\phi \in \mathscr{F}^{\mathcal{D}}_{A(\Sigma),X}$ and postcondition $\psi \in \mathscr{F}^{2\mathcal{D}}_{A(\Sigma),X \cup \bigcup_{1 \leq k \leq m} X_k}$
- an initial control state $c_0 \in C$ and state predicate $\varphi_0 \in \mathscr{F}^{\mathcal{D}}_{A(\Sigma),\emptyset}$

where all C syntactically reachable from c_0

Completion events captured by extending the signature Σ by input events

Example



$(\mathit{PINEntered}, \mathrm{true}, \text{PINEntered}, \text{verify}(x_1, x_2), x_1 = \text{cardId} \wedge x_2 = \text{pin}, \mathit{Verifying})$

Simple UML State Machines (2)

Loose semantics of $U = (\Sigma, C, T, (c_0, \varphi_0))$ given by model class of event/data transition structures with $M \in \mathrm{Mod}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}(U)$ if

- $R(M)$ only shows transitions allowed by T
  - for all $((c, d), (c', d')) \in R(M)_{i(\beta), O(\beta')}$ there is a $(c, \phi, i, O, \psi, c') \in T$ such that $\omega(M)(d), \beta \models^{\mathcal{D}}_{A(\Sigma), X(i)} \phi$ and $(\omega(M)(d), \omega(M)(d')), \beta \cup \beta' \models^{2\mathcal{D}}_{A(\Sigma), X(\{i\} \cup O)} \psi$
- $R(M)$ realises T for satisfied preconditions
  - for all $(c, \phi, i, O, \psi, c') \in T$ and $\beta \colon X(i) \to \mathcal{D}$ with $\omega(M)(d), \beta \models^{\mathcal{D}}_{A(\Sigma), X(i)} \phi$, there is a $\beta' \colon X(O) \to \mathcal{D}$ and a $((c, d), (c', d')) \in R(M)_{i(\beta), O(\beta')}$ such that $(\omega(M)(d), \omega(M)(d')), \beta \cup \beta' \models^{2\mathcal{D}}_{A(\Sigma), X(i) \cup X(O)} \psi$

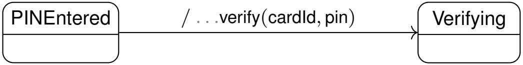
Input enabledness handled by self-loops for events to be discarded

Characterising Simple UML State Machines in $\mathcal{M}^{\downarrow}_{\mathcal{D}}$

Theorem For every simple UML state machine U there is an $\mathcal{M}^{\downarrow}_{\mathcal{D}}$-sentence ϱ_U such that

$$M \in \mathrm{Mod}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}(U) \iff M \models^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma(U)} \varrho_U \; .$$

Example



$$\begin{aligned}(@\mathit{PINEntered})\langle\!|\, &\text{PINEntered} : \mathrm{true} /\!\!/ \text{verify}(x_1, x_2) : x_1 = \text{cardId} \wedge x_2 = \text{pin} \,|\!\rangle \mathit{Verifying} \wedge \\ &[\text{PINEntered} /\!\!/ \text{verify}(x_1, x_2) : x_1 \neq \text{cardId} \vee x_2 \neq \text{pin}]\mathrm{false} \wedge \\ &[\text{PINEntered} /\!\!/ o : \mathrm{true}]\mathrm{false} \wedge \\ &[\text{PINEntered} /\!\!/ [O']_{O(\Sigma_{ATM})} : \mathrm{true}]\mathrm{false}\end{aligned}$$

with $o \neq \text{verify}(x_1, x_2)$ and $O' = o_1\, o_2$

Alternative: direct characterisation of trans for reachable configurations

Simple UML Composite Structures (1)

Composite structure $\Delta = (\mathit{Cmp}, \mathit{Prt}, \mathit{Con})$

- components (parts) $c \in \mathit{Cmp}$ with $\mathcal{M}^{\downarrow}_{\mathcal{D}}$-signature $\Sigma(c)$ for input/output events and attributes
- ports $p \in \mathit{Prt}$ of component $\mathit{cmp}(p) \in \mathit{Cmp}$ and $\mathcal{M}^{\downarrow}_{\mathcal{D}}$-signature $\Sigma(p)$ for provided (input) and required (output) events
- connectors $\mathit{Con} \subseteq \mathit{Prt} \times \mathit{Prt}$
    - port open if not connected

Example

Directly taken to be $\mathrm{CS}(\mathcal{M}^{\downarrow}_{\mathcal{D}})$-signatures $\mathbb{S}^{\mathrm{CS}(\mathcal{M}^{\downarrow}_{\mathcal{D}})}$

Δ-composite structure structures $\mathit{Str}^{\mathrm{CS}(\mathcal{M}^{\downarrow}_{\mathcal{D}})}(\Delta)$

$$\mathscr{C} \in (\mathscr{C}(c) \in |\mathit{Str}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}(\Sigma(\Delta, c))|)_{c \in \mathit{Cmp}(\Delta)}$$

Simple UML Composite Structures (2)

Integration of event pools by embedding into $\mathcal{M}^{\downarrow}_{\mathcal{D}}$

- extension of $\mathsf{cs}(\mathcal{M}^{\downarrow}_{\mathcal{D}})$-signature Δ to a queue-based $\mathcal{M}^{\downarrow}_{\mathcal{D}}$-signature $\Sigma_{\mathrm{q}}(\Delta)$ by event queue attributes q_c

$$\Sigma_{\mathrm{q}}(\Delta) = \bigcup_{c \in \mathit{Cmp}(\Delta)} (\Sigma(\Delta, c) \cup \{\mathrm{q}_c : \hat{I}(\Sigma(\Delta, c))^*\})$$

- extension of $\mathsf{cs}(\mathcal{M}^{\downarrow}_{\mathcal{D}})$-structure $\mathscr{C}$ to a $\Sigma_{\mathrm{q}}(\Delta)$-event/data structure $M_{\mathscr{C}}$
  - configurations $(q(c), \gamma(c))_{c \in \mathit{Cmp}(\Delta)}$ of event queues $q(c) \in \hat{I}(\Sigma(\Delta, c))^*$ stored in q_c and part configurations $\gamma(c) \in \Gamma(\mathscr{C}(c))$
  - transitions by selecting an event from a part's event queue, letting the part react to this event, and distributing the produced messages
- formulæ of $\mathsf{cs}(\mathcal{M}^{\downarrow}_{\mathcal{D}})$ formulæ of $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ over $\mathcal{D}$ and queues

$$\mathscr{C} \models^{\mathsf{cs}(\mathcal{M}^{\downarrow}_{\mathcal{D}})}_{\Delta} \varrho \iff M_{\mathscr{C}} \models^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma_{\mathrm{q}}(\Delta)} \varrho$$

Institution $(\mathbb{S}^{\mathsf{cs}(\mathcal{M}^{\downarrow}_{\mathcal{D}})}, \mathit{Str}^{\mathsf{cs}(\mathcal{M}^{\downarrow}_{\mathcal{D}})}, \mathrm{Sen}^{\mathsf{cs}(\mathcal{M}^{\downarrow}_{\mathcal{D}})}, \models^{\mathsf{cs}(\mathcal{M}^{\downarrow}_{\mathcal{D}})})$

Example Verification

Little support for induction in fully automatic CASL-provers connected to HETS

- manually crafted induction schemes for all generated datatypes needed

Resort to interactive theorem prover KIV

- support for algebraic specifications similar to CASL
- rich heuristics for induction
- connection to HETS under development

Example Verification in KIV (1)

ATM and bank component transition systems (automatic)

```
atmTrans-def: atmTrans(atmConf(sa1,c1,p1,t1), in, out, atmConf(sa2,c2,p2,t2))
  ↔ ∃ c : CardId, p : Pin . ...
        ∨ (   sa1 = CardEntered
           ∧ in = msg(userCom, PIN(p)) ∧ out = (msg(atmCompl, PINEnteredCompl) +
           ∧ p2 = p ∧ sa2 = PINEntered ∧ c2 = c1 ∧ t2 = t1)
        ∨ ...; used for: s, ls;
```

Composite structure transition system (automatic)

```
trans-def: trans(conf(ca1,qa1,cb1,qb1), in, out, conf(ca2,qa2,cb2,qb2))
  ↔   dist(out, qa1, qa2, qb1, qb2)
     ∧ (   (atmTrans(ca1, in, out, ca2) ∧ cb2 = cb1 )
        ∨ (bankTrans(cb1, in, out, cb2) ∧ ca2 = ca1)); used for: s, ls;
```

Safety predicate (user input)

```
safe-def: safe(g) ↔ (ctrl(caConf(g)) = Verified → wasVerified(cbConf(g)) = 1);
used for: s, ls;
```

Example Verification in KIV (2)

Generalising invariant (user input)

```
invar-def: invar(conf(ca, qa, cb, qb)) ↔ ∃ x.
    (ctrl(ca) = Idle        ∧ ctrl(cb) = Idle ∧ qa = empty ∧ qb = empty)
  ∨ (ctrl(ca) = CardEntered ∧ ctrl(cb) = Idle ∧ qa = empty ∧ qb = empty)
  ∨ (ctrl(ca) = PINEntered  ∧ ctrl(cb) = Idle ∧ qa = enq(x, empty) ∧ qb = empty)
  ...; used for: s, ls;
```

Proof obligations (automatic)

```
Safe: invar(g) → safe(g);
Init: init(g) → invar(g);
...
Trans6: g1 = conf(atmConf(Verifying, c, p, t), qa, cb, qb)
      ∧ qa ≠ empty ∧ top(qa) = msg(atmCom, verified)
      ∧ g2 = conf(atmConf(Verified, c, p, t),
                  enq(msg(atmCompl, VerifiedCompl), deq(qa)), cb, qb)
      ∧ invar(g1) → invar(g2);
```

All proof obligations discharged automatically by KIV heuristics

Conclusions and Future Work

Heterogeneous modelling

- institution of event/data-based hybrid modal logic $\mathcal{M}^{\downarrow}_{\mathcal{D}}$
  - basis for simple UML state machines
  - basis for UML composite structures
- theoroidal institution comorphism from $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ to CASL
  - basis for theorem proving

- lemma generation for proof automation
- revisit relation to UML interactions
- include other languages like TLA or Event-B