Specifying Event/Data-based Systems

Alexander Knapp

Universität Augsburg

Joint work with Rolf Hennicker and Alexandre Madeira

Specifying Event/Data-based Systems

Event/Data-based systems

- behaviour controlled by events
- data states may change in reaction to events

Specification of event/data-based systems

- Model-oriented approaches (constructive specification)
  - Event-B, symbolic transition systems, CSP with data, UML behavioural/protocol state machines
- Property-oriented approaches (abstract specification)
  - modal (temporal, dynamic) logics, TLA
- Checking whether a concrete model satisfies certain abstract properties
- Refining abstract models to concrete implementations

Example: Specifying an ATM

Events $\{\text{insertCard}, \text{enterPIN}, \text{ejectCard}\}$, data attributes $\{\text{chk}\}$

Axiomatic specification using modal logic formulæ, like

- "Whenever a card has been inserted, a correct PIN can eventually be entered."
  - $\mathbf{AG}(\text{done}(\text{insertCard}) \rightarrow \mathbf{EF}(\text{done}(\text{enterPIN}) \wedge \text{chk} = tt))$
  - $[\boldsymbol{E}^*; \text{insertCard}]\langle \boldsymbol{E}^*; (\text{enterPIN} /\!\!/ \text{chk}' = tt)\rangle \text{true}$

- "Whenever a correct PIN has been entered, the card can eventually be ejected."
  - $\mathbf{AG}(\text{chk} = tt \rightarrow \mathbf{EF}\,\text{enabled}(\text{ejectCard}))$
  - $[\boldsymbol{E}^*; (\text{enterPIN} /\!\!/ \text{chk}' = tt)]\langle \boldsymbol{E}^*; \text{ejectCard}\rangle \text{true}$

- "A card cannot be ejected if it has not been inserted before."
  - $\neg\text{enabled}(\text{ejectCard})\ \mathbf{W}\ \text{done}(\text{insertCard})$
  - $[(-\text{insertCard})^*; \text{ejectCard}]\text{false}$

Properties in mCRL2 (1)

Jan Friso Groote, Mohammad Reza Mousavi. Modeling and Analysis of Communicating Systems, 2014

- modal μ-calculus for transition systems specified in process algebra
- based on abstract data types and multi-actions for events, no special notion of states

$$
\begin{aligned}
&\nu ATM(c : State := \mathsf{Card}, chk : \mathbb{B} := ff, trls : \mathbb{N} := 0)\,. \\
&\quad c = \mathsf{Card} \to \langle \mathsf{insertCard} \rangle ATM(\mathsf{PIN}, ff, 0) \\
&\wedge\, c = \mathsf{PIN} \to \langle \mathsf{enterPIN} \rangle (\quad trls < 2 \to ATM(\mathsf{PIN}, ff, trls + 1) \vee \\
&\qquad\qquad\qquad\qquad\qquad\qquad ATM(\mathsf{Return}, tt, trls + 1) \\
&\qquad\qquad\qquad\qquad \wedge\, trls = 2 \to ATM(\mathsf{Card}, ff, trls + 1) \vee \\
&\qquad\qquad\qquad\qquad\qquad\qquad ATM(\mathsf{Return}, tt, trls + 1)) \\
&\wedge\, c = \mathsf{PIN} \to \langle \mathsf{cancel} \rangle ATM(\mathsf{Return}, ff, trls) \\
&\wedge\, c = \mathsf{Return} \to \langle \mathsf{ejectCard} \rangle ATM(\mathsf{Card}, chk, trls)
\end{aligned}
$$

Properties in mCRL2 (2)

Integrates (multi-)actions, data, regular formulæ, and modal formulæ

$$
\begin{aligned}
\alpha &::= \tau \mid a(t_1, \ldots, t_n) \mid \alpha_1 \mid \alpha_2 \\
\eta &::= b \mid \text{false} \mid \alpha \mid \overline{\eta} \mid \eta_1 \cup \eta_2 \mid \exists d : D \,.\, \eta \\
\rho &::= \varepsilon \mid \eta \mid \rho_1 \cdot \rho_2 \mid \rho_1 + \rho_2 \mid \rho^* \\
\phi &::= b \mid \text{false} \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \exists d : D \,.\, \phi \mid \\
&\quad \langle\rho\rangle\phi \mid \mu X(d_1 : D_1 := t_1, \ldots, d_n : D_n := t_n) \,.\, \phi \mid X(t_1, \ldots, t_n)
\end{aligned}
$$

- Very expressive
  - fixed point operators for safety and liveness, may also be mixed

    $$\mu X \,.\, \nu Y \,.\, (\langle a \rangle \text{true} \wedge [b] X) \vee ([a]\text{false} \wedge [b] Y)$$

  - fixed point variables may have parameters

    $$\nu X(n : \mathbb{N} := 0) \,.\, [\overline{d \cup r}] X(n) \wedge [r] X(n+1) \wedge [d](n > 0 \wedge X(n-1))$$

- rich tool support for model checking and simulation

Temporal Logic of Actions (1)

Leslie Lamport. Specifying Systems. Addison Wesley, 2002

- linear temporal logic for describing transition systems
- data from general set theory, no special notions of states or events

$$
\begin{aligned}
InitATM &\equiv st = Card \land chk \in \mathbb{B} \land trls \in \mathbb{N} \\
InsertCard &\equiv st = Card \land chk' = ff \land trls' = 0 \land st' = PIN \\
EnterPIN1 &\equiv st = PIN \land trls < 2 \land chk' = ff \land trls' = trls + 1 \land st' = PIN \\
EnterPIN2 &\equiv st = PIN \land trls = 2 \land chk' = ff \land trls' = trls + 1 \land st' = Card \\
EnterPIN3 &\equiv st = PIN \land trls \leq 2 \land chk' = tt \land trls' = trls + 1 \land st' = Return \\
EnterPIN &\equiv EnterPIN1 \lor EnterPIN2 \lor EnterPIN3 \\
Cancel &\equiv st = PIN \land chk' = ff \land trls' = trls \land st = Return \\
EjectCard &\equiv st = Return \land chk' = chk \land trls' = trls \land st = Card \\
NextATM &\equiv InsertCard \lor EnterPIN \lor Cancel \lor EjectCard \\
ATM &\equiv InitATM \land \Box[NextATM]_{st,chk,trls} \land \mathrm{WF}_{st,chk,trls}(NextATM)
\end{aligned}
$$

Temporal Logic of Actions (2)

General TLA system specification format

$$Init \wedge \Box[Next]_{\vec{v}} \wedge L$$

- $Init$ state predicate for initial states
- $Next$ disjunction of transition predicates (actions)
  - involving primed and unprimed variables
- tuple $\vec{v}$ of variables that can be changed by the system
  - $[A]_{v_1,\ldots,v_n} \equiv A \vee (v_1 = v_1' \wedge \ldots \wedge v_n = v_n')$
- L conjunction of action fairness conditions (weak, strong)

TLA stutter-invariant

- stuttering steps without changes to system variables
- parallel composition (mainly) by conjunction, trace refinement as implication
- property specifications in LTL

Alloy (1)

Daniel Jackson. Software Abstractions. MIT Press, 2016

- relational specification of states, data, and events
- since Alloy 6 support for transition systems and linear temporal logic

```
enum State { Card, PIN, Return }
one sig ATM { var st : State, var chk : Bool, var trls : Int }

pred init[a : ATM] { a.st = Card && a.chk = False && a.trls = 0 }

pred insertCard[a : ATM] { a.st = Card
  a.chk' = False && a.trls' = 0 && a.st' = PIN }

pred enterPIN[a : ATM] { a.st = PIN
  a.trls' = plus[a.trls, 1] &&
  (  (a.trls < 2 && a.chk' = False && a.st' = PIN)
  || (a.trls = 2 && a.chk' = False && a.st' = Card)
  || (a.trls <= 2 && a.chk' = True && a.st' = Return)) }

pred cancel[a : ATM] { a.st = PIN
  a.chk' = False && a.trls' = a.trls && a.st' = Return }

pred ejectCard[a : ATM] { a.st = Return
  a.chk' = a.chk && a.trls' = a.trls && a.st' = Card }
```

Alloy (2)

- Alloy 6 based on Electrum (2016ff.), similar to TLA$^+$
  - Chris Newcombe. Why Amazon Chose TLA$^+$. ABZ 2014
- Mutable fields declared by `var`

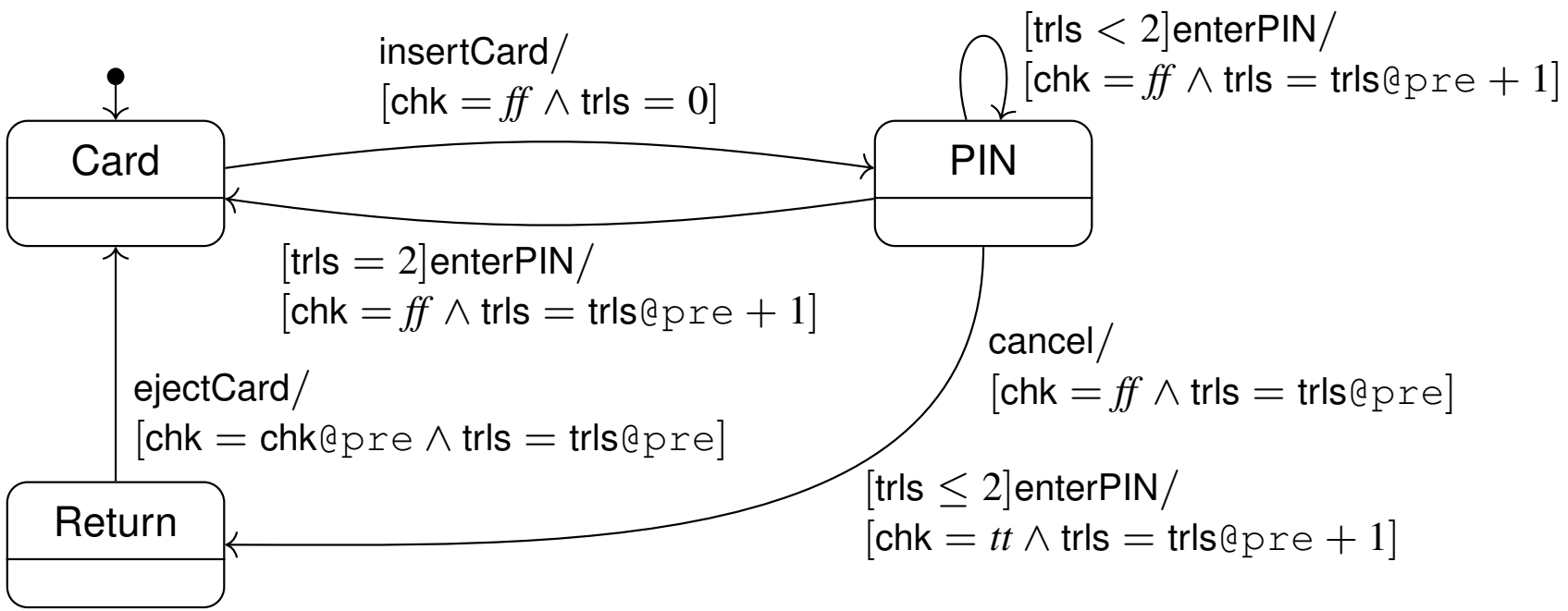
- Bounded model finding and visualisation
  - runs for LTL always are loops

```
run {
  some atm : ATM {
    init[atm]
    always (  insertCard[atm] || enterPIN[atm] || cancel[atm]
           || ejectCard[atm] || atm.trls >= 3)
    eventually atm.chk = True
  }
} for 3..10 steps
```

UML State Machines (1)

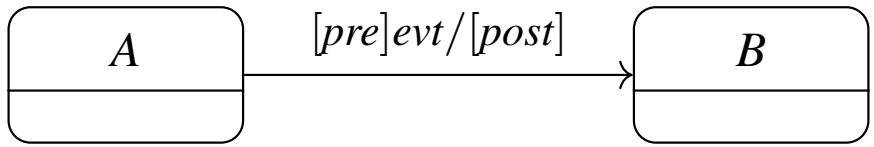
Object Managment Group. Unified Modeling Language 2.5.1. formal/2017-12-05, 2017

- protocol and behaviour state machines (like Harel state charts)
- data from contextual static structure, dedicated support for (hierarchical) states and (signal, call, &c.) events

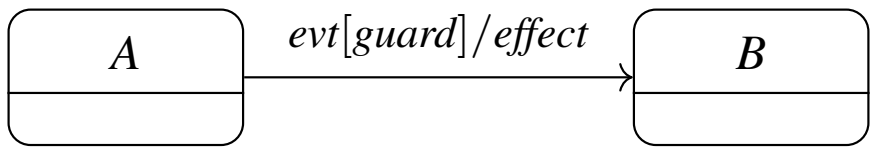


UML State Machines (2)

Protocol state machines for describing legal sequences of event occurrences



Behaviour state machines for operational specifications



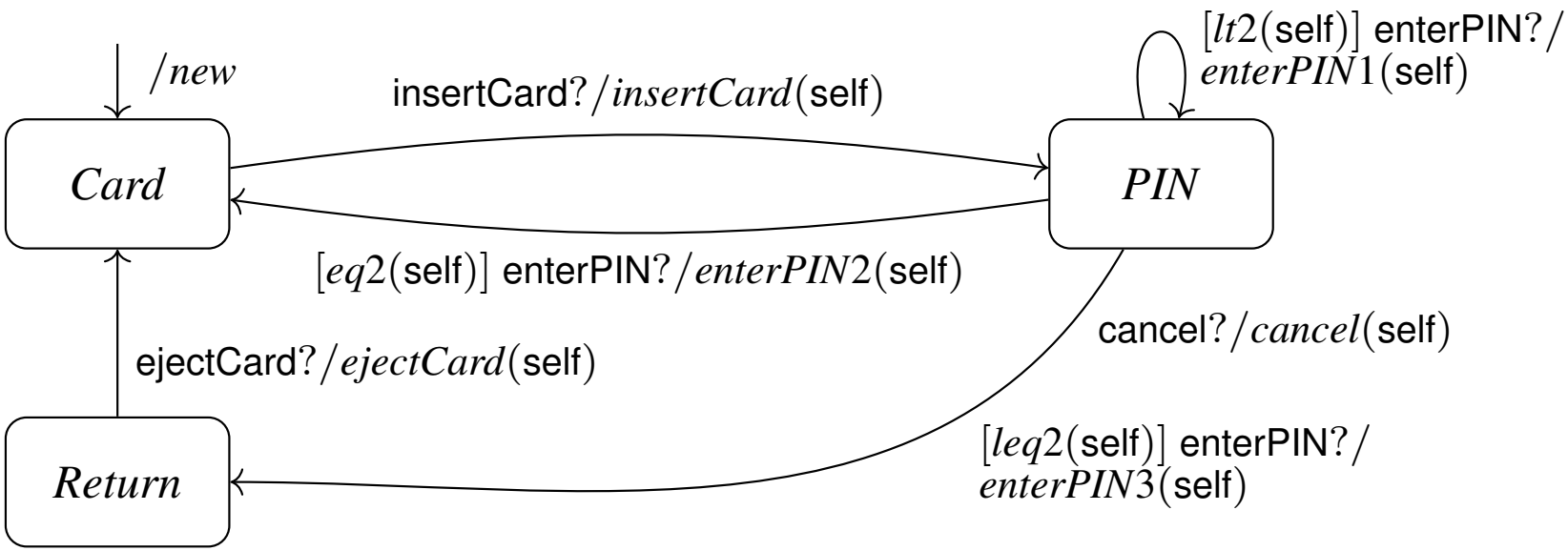
UML notorious for lack of semantics

- parallel composition "synchronously" by orthogonal regions (same machine), or "asynchronously" (different machines) via event queues
- refinement ("redefinition") rather unclear
- property specifications in OCL (`oclInState`)

Symbolic Transition Systems (1)

Pascal Poizat, Jean-Claude Royer. A Formal Architectural Description Language based on Symbolic Transition Systems and Modal Logic. J. Univ. Comp. Sci. 12(12), 2006, pp. 1741–1782

- symbolic analysis by unfolding
- data from underlying abstract data type, explicit states and events



$$insertCard(newATM(chk, trls)) = newATM(ff, 0)$$
$$lt2(newATM(chk, trls)) = trls < 2$$

Symbolic Transition Systems (2)

General form of transitions

$$T \subseteq S \times Tm(\Sigma_{Bool}, X) \times Evt(L) \times Tm(\Sigma_{Dt}, X) \times S$$

- source (control) state, guard term, event (input/output), action term, target (control) state
- Unfolding into symbolic state set S': if $v \in Tm(\Sigma_{Dt})$, $(s, v) \in S'$, $(s, g, e, a, t) \in T$, and $g(v)\downarrow$, then $(t, a(v)\downarrow) \in S'$
- initial semantics of abstract data type

Used for specifying software architectures

- parallel composition by synchronous product
- property specifications in modal logic over events with (control) state test $@_s$ and state binding $\overset{@}{\exists} x \,.\, \varphi$

Communicating Sequential Processes (1)

Andrew W. Roscoe. The Theory and Practice of Concurrency. Prentice Hall, 1998

- process algebra (like CCS, LOTOS, μCRL, mCRL2)
- data from set theory, process terms as states, labels as events

$$\begin{aligned}
\mathsf{Card}(chk, trls) &= \mathsf{insertCard} \rightarrow \mathsf{PIN}(\mathit{ff}, 0) \\
\mathsf{PIN}(chk, trls) &= (trls < 2 \;\&\; \mathsf{enterPIN} \rightarrow \mathsf{PIN}(\mathit{ff}, trls + 1) \\
&\qquad \sqcap \mathsf{Return}(\mathit{tt}, trls + 1)) \\
&\quad \Box\ (trls = 2 \;\&\; \mathsf{enterPIN} \rightarrow \mathsf{Card}(\mathit{ff}, trls + 1) \\
&\qquad \sqcap \mathsf{Return}(\mathit{tt}, trls + 1)) \\
&\quad \Box\ (\mathsf{cancel} \rightarrow \mathsf{Return}(\mathit{ff}, trls)) \\
\mathsf{Return}(chk, trls) &= \mathsf{ejectCard} \rightarrow \mathsf{Card}(chk, trls) \\
\mathsf{ATM} &= \sqcap_{(chk,trls) \in \mathbb{B} \times \mathbb{N}} \mathsf{Card}(chk, trls)
\end{aligned}$$

Communicating Sequential Processes (2)

Semantics based on traces, failures, and divergences

$$tr(a \to P) = \{\langle\rangle\} \cup \{\langle a\rangle \frown s \mid s \in tr(P)\}$$

$$tr(c \;\&\; P) = \begin{cases} tr(P) & \text{if } c \text{ evaluates to true} \\ \{\langle\rangle\} & \text{otherwise} \end{cases}$$

$$tr(P \,\Box\, Q) = tr(P) \cup tr(Q)$$

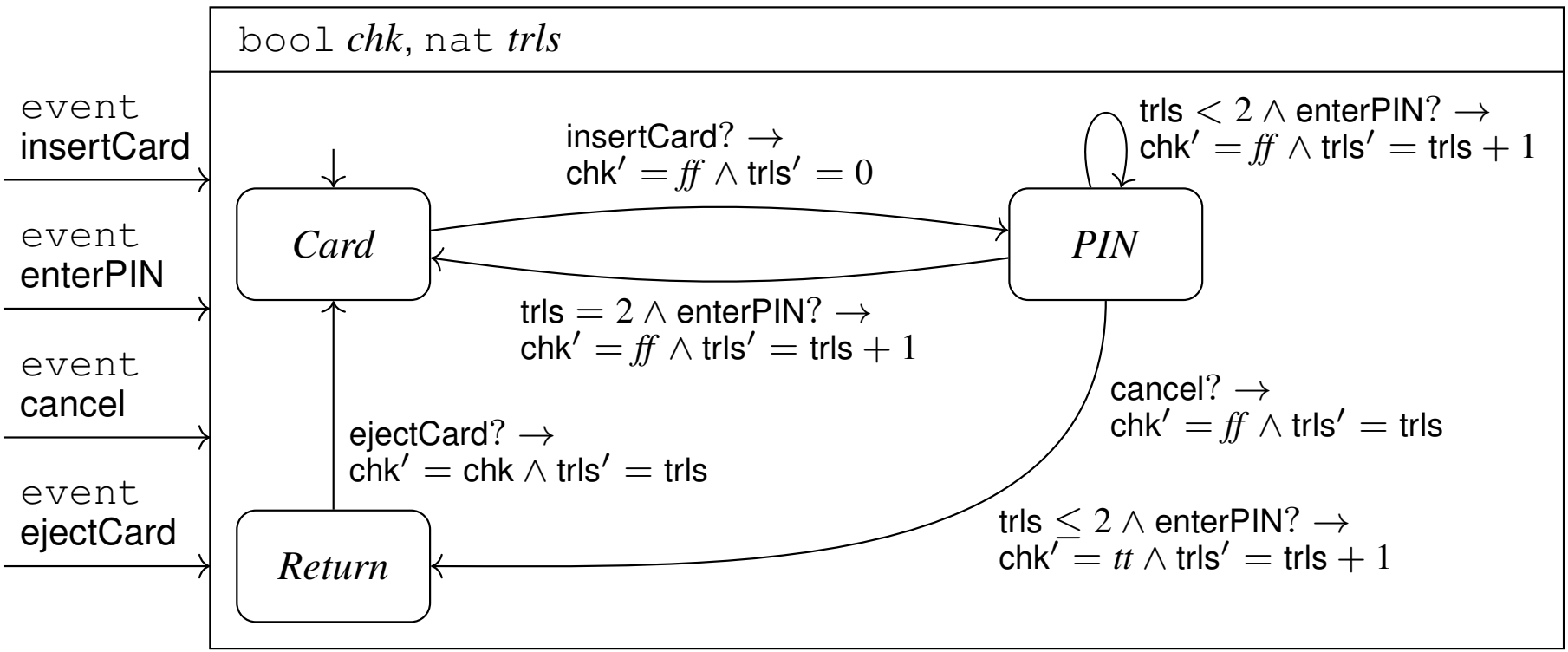
$$tr(P \sqcap Q) = tr(P) \cup tr(Q)$$

- (synchronous) parallel composition integral part of the language
- property specifications also given by CSP processes
- combination with algebraic specification language CASL for loose semantics of data

Synchronous Languages (1)

Rajeev Alur. Principles of Cyber-Physical Systems. MIT Press, 2015

- tick-based, synchronous execution (like Lustre, Esterel)
- events given by status of signals (present or absent)



Synchronous Languages (2)

Synchrony hypothesis

- Model of computation: In each tick, read inputs, compute, produce outputs
  - synchronous and instanteneous over all tasks
  - dependency checks over tasks
- Semantics: $s_1 \xrightarrow{i_1/o_1} s_2 \xrightarrow{i_2/o_2} \cdots$

Status (and values) of all signals form a single event

- parallel composition by task union
- asynchronous variant based on channels (similar to Promela)

Event-B (1)

Jean-Raymond Abrial. Modeling in Event-B. Cambridge University Press, 2010

- abstract state machine approach (like Z, B)
- events as transitions, state and data from set theory

$$\textbf{invariant}\ st \in \{Card, PIN, Return\} \land chk \in \mathbb{B} \land trls \in \mathbb{N}$$

$$\textbf{events}\ \text{init} \mathrel{\widehat{=}} \textbf{then}\ st := Card\ \textbf{end}$$

$$\text{insertCard} \mathrel{\widehat{=}} \textbf{when}\ st = Card\ \textbf{then}\ chk := ff;\ trls := 0; st := PIN\ \textbf{end}$$

$$\begin{aligned}\text{enterPIN} \mathrel{\widehat{=}} \textbf{when}\ & st = PIN\ \textbf{then}\ chk, trls, st :| \\ & (trls < 2 \land chk' = ff \land trls' = trls + 1 \land st' = PIN) \lor \\ & (trls = 2 \land chk' = ff \land trls' = trls + 1 \land st' = Card) \lor \\ & (trls \leq 2 \land chk' = tt \land trls' = trls + 1 \land st' = Return) \\ & \textbf{end}\end{aligned}$$

$$\text{cancel} \mathrel{\widehat{=}} \textbf{when}\ st = PIN\ \textbf{then}\ chk := ff; st := Return\ \textbf{end}$$

$$\text{ejectCard} \mathrel{\widehat{=}} \textbf{when}\ st = Return\ \textbf{then}\ st := Card\ \textbf{end}$$

Event-B (2)

General event format

$$e \mathrel{\widehat{=}} \textbf{status}\ \sigma\ \textbf{when}\ G(\vec{v})\ \textbf{then}\ \vec{v} :\mid H(\vec{v}, \vec{v}')$$

- guard G, before-after predicate H (or assignments)
- status ‘ordinary’ or ‘convergent’ (internal: decreasing variant) or ‘anticipated’ (not increasing variant)

Focus on (data) refinement (proof obligations)

- traces based on weakest precondition semantics
- introducing new events by refining $skip$

- Parallel (de-)composition can be added (M. Butler 2009)
- Combination with CSP for explicit control (S. Schneider, H. Treharne 2010)

Specifying Event/Data-based Systems Revisited

Goals

- Common logical formalism for specifying event/data-based systems on various levels of abstraction
- Program development by stepwise refinement ("correct by construction")
  - from axiomatic to operational specifications
  - based on rigorous formal semantics

Approach — $\mathcal{E}^{\downarrow}$

- Integrate dynamic and hybrid logic features
  - Dynamic logic for abstract requirements (safety, liveness, ...)
  - Hybrid logic for concrete process structure
- Apply Sannella & Tarlecki's refinement methodology in the context of event/data-based systems

Rolf Hennicker, Alexandre Madeira. A. K. A Hybrid Dynamic Logic for Event/Data-based Systems. FASE 2019

Syntax: Event/Data Actions and Formulæ (1)

Ed signature $\Sigma = (E, A)$ with events E and data attributes A

- data state $\omega \in \Omega(\Sigma) = A \to \mathcal{D}$
- state predicates $\varphi \in \Phi(\Sigma)$ with $\omega \models_A^{\mathcal{D}} \varphi$
- transition predicates $\psi \in \Psi(\Sigma)$ with $(\omega, \omega') \models_A^{\mathcal{D}} \psi$

Σ-ed actions $\lambda \in \Lambda(\Sigma)$

$$\lambda ::= e /\!\!/ \psi \mid \lambda_1 + \lambda_2 \mid \lambda_1 ; \lambda_2 \mid \lambda^*$$

- transition specification $e /\!\!/ \psi$ for event e and effect specification ψ
  - abbreviate $e_1 /\!\!/ \mathrm{true} + \ldots + e_k /\!\!/ \mathrm{true}$ by $\{e_1, \ldots, e_k\}$, $E(\Sigma)$ by $\boldsymbol{E}$, ...
- complex actions with "or" $+$, "sequence" ;, and "iteration" *

Example: $\boldsymbol{E}^* ; \mathsf{enterPIN} /\!\!/ \mathsf{chk}' = tt$

"a finite sequence of events with arbitrary effects followed by event enterPIN such that afterwards attribute chk is tt"

Syntax: Event/Data Actions and Formulæ (2)

Σ-ed formulæ $\varrho \in \mathrm{Frm}^{\mathcal{E}^{\downarrow}}(\Sigma)$

$$\varrho ::= \varphi \mid x \mid \downarrow x \,.\, \varrho \mid (@x)\varrho \mid \langle\lambda\rangle\varrho \mid [\lambda]\varrho \mid \mathrm{true} \mid \neg\varrho \mid \varrho_1 \vee \varrho_2$$

- state predicates φ
- control state variables $x \in X$
- hybrid logic "bind" $\downarrow x$ and "jump" $@x$
- dynamic logic "diamond" $\langle\lambda\rangle$ and "box" $[\lambda]$
- usual propositional connectives

Example: $\downarrow x_0 \,.\, [\boldsymbol{E}^*; (\mathsf{enterPIN} /\!\!/ \mathsf{chk}' = tt) + \mathsf{cancel}] \langle \boldsymbol{E}^*; \mathsf{ejectCard} \rangle x_0$

"Whenever a correct PIN has been entered or the transaction has been cancelled, the card can eventually be ejected and the ATM starts from the beginning."

Semantics: Event/Data Transition Systems

Σ-edts $M = (\Gamma, R, \Gamma_0) \in Edts^{\mathcal{E}^\downarrow}(\Sigma)$

- configurations $\Gamma \subseteq C \times \Omega(\Sigma)$ of control states C and data states $\Omega(\Sigma)$
- transition relations $R \subseteq (R_e \subseteq \Gamma \times \Gamma)_{e \in E(\Sigma)}$
- initial configurations $\Gamma_0 \subseteq \{c_0\} \times \Omega_0$ with $\Omega_0 \subseteq \Omega(\Sigma)$
  - all configurations required to be reachable

Interpretation of Σ-ed actions over M as $(R_\lambda \subseteq \Gamma \times \Gamma)_{\lambda \in \Lambda(\Sigma)}$ defined by

- $R_{e/\!\!/\psi} = \{((c,\omega),(c',\omega')) \in R_e \mid (\omega,\omega') \models^{\mathcal{D}}_{A(\Sigma)} \psi\}$
- $R_{\lambda_1+\lambda_2} = R_{\lambda_1} \cup R_{\lambda_2}$
- $R_{\lambda_1;\lambda_2} = R_{\lambda_1}; R_{\lambda_2}$
- $R_{\lambda^*} = (R_\lambda)^*$

Semantics: Event/Data Satisfaction Relation

Given Σ-edts M, valuation $v : X \to C(M)$, configuration $\gamma \in \Gamma(M)$

- $M, v, \gamma \models^{\mathcal{E}^\downarrow}_\Sigma \varphi$ iff $\omega(\gamma) \models^{\mathcal{D}}_{A(\Sigma)} \varphi$
- $M, v, \gamma \models^{\mathcal{E}^\downarrow}_\Sigma x$ iff $c(\gamma) = v(x)$
- $M, v, \gamma \models^{\mathcal{E}^\downarrow}_\Sigma \downarrow x \,.\, \varrho$ iff $M, v\{x \mapsto c(\gamma)\}, \gamma \models^{\mathcal{E}^\downarrow}_\Sigma \varrho$
- $M, v, \gamma \models^{\mathcal{E}^\downarrow}_\Sigma (@x)\varrho$ iff $M, v, \gamma' \models^{\mathcal{E}^\downarrow}_\Sigma \varrho$ for all $\gamma' \in \Gamma(M)$ with $c(\gamma') = v(x)$
- $M, v, \gamma \models^{\mathcal{E}^\downarrow}_\Sigma \langle\lambda\rangle\varrho$ iff $M, v, \gamma' \models^{\mathcal{E}^\downarrow}_\Sigma \varrho$ for some $\gamma' \in \Gamma(M)$ with $(\gamma, \gamma') \in R(M)_\lambda$
- ...

$M \models^{\mathcal{E}^\downarrow}_\Sigma \varrho$ for Σ-ed sentences if $M, v, \gamma_0 \models^{\mathcal{E}^\downarrow}_\Sigma \varrho$ for all $\gamma_0 \in \Gamma_0(M)$

Axiomatic Event/Data Specifications

Axiomatic ed specification $Sp = (\Sigma, Ax)$ over ed signature Σ

- set of Σ-ed sentences Ax as axioms

(Loose) semantics of Sp given by model class $\mathrm{Mod}(Sp)$ of edts

- $\mathrm{Mod}(Sp) = \{M \in Edts^{\mathcal{E}^{\downarrow}}(\Sigma) \mid M \models^{\mathcal{E}^{\downarrow}}_{\Sigma} Ax\}$

Example: Specification ATM_0 with $\Sigma_0 = (\{\mathsf{insertCard}, \ldots\}, \{\mathsf{chk}\})$ and Ax_0:

$$[\boldsymbol{E}^*; (\mathsf{enterPIN} /\!\!/ \mathsf{chk}' = tt) + \mathsf{cancel}] \langle \boldsymbol{E}^*; \mathsf{ejectCard} \rangle \mathrm{true} \ , \quad \ldots$$

Example: Specification ATM_1 with $\Sigma_1 = \Sigma_0$ and Ax_1:

$$\downarrow x_0 \,.\, [\boldsymbol{E}^*; (\mathsf{enterPIN} /\!\!/ \mathsf{chk}' = tt) + \mathsf{cancel}] \langle \boldsymbol{E}^*; \mathsf{ejectCard} \rangle x_0$$

$$\downarrow x_0 \,.\, \langle \mathsf{insertCard} /\!\!/ \mathsf{chk}' = ff \rangle \mathrm{true} \wedge [\mathsf{insertCard} /\!\!/ \neg(\mathsf{chk}' = ff)] \mathrm{false} \wedge [-\mathsf{insertCard}] \mathrm{false} \ , \quad \ldots$$

Stepwise refinement in $\mathcal{E}^{\downarrow}$: $ATM_0 \rightsquigarrow ATM_1 \rightsquigarrow \ldots$

Refining $\mathcal{E}^{\downarrow}$-Specifications (1)

Simple refinement (or implementation) relation for specifications

$$Sp \rightsquigarrow Sp' \text{ if } \Sigma(Sp) = \Sigma(Sp') \text{ and } \mathrm{Mod}(Sp) \supseteq \mathrm{Mod}(Sp')$$

- no signature changes, no construction of an implementation

Constructor κ from $(\Sigma'_1, \ldots, \Sigma'_n)$ to Σ

- (total) function $\kappa : Edts^{\mathcal{E}^{\downarrow}}(\Sigma'_1) \times \ldots \times Edts^{\mathcal{E}^{\downarrow}}(\Sigma'_n) \to Edts^{\mathcal{E}^{\downarrow}}(\Sigma)$
- constructor composition by usual function composition

$\langle Sp'_1, \ldots, Sp'_n \rangle$ constructor implementation via κ of Sp

- $Sp \rightsquigarrow_{\kappa} \langle Sp'_1, \ldots, Sp'_n \rangle$ if $\kappa(M'_1, \ldots, M'_n) \in \mathrm{Mod}(Sp)$ for all $M'_i \in \mathrm{Mod}(Sp'_i)$

(Sannella, Tarlecki 1988)

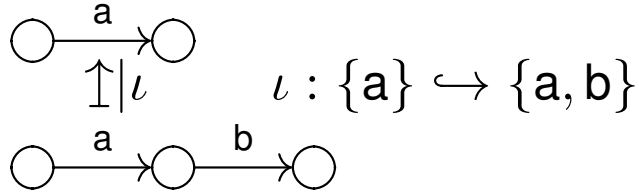
Refining $\mathcal{E}^{\downarrow}$-Specifications (2)

Refinement chain

$$Sp_1 \rightsquigarrow_{\kappa_1} Sp_2 \rightsquigarrow_{\kappa_2} \cdots \rightsquigarrow_{\kappa_{n-1}} Sp_n$$

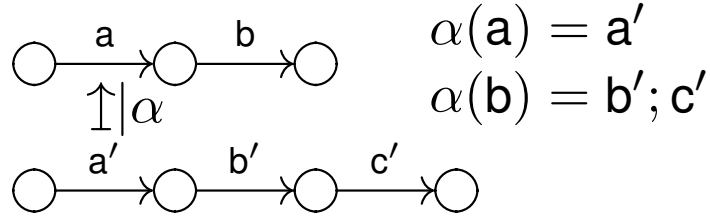
Constructors for $\mathcal{E}^{\downarrow}$-specifications

- relabelling κ_ρ
  - ρ-reduct of edts for a bijective ed signature morphism ρ
- restriction κ_ι
  - ι-reduct of edts for an injective ed signature morphism ι



- event refinement κ_α
  - α-reduct of edts for an ed signature morphism α to composite events

$$\theta ::= e \mid \theta + \theta \mid \theta; \theta \mid \theta^*$$



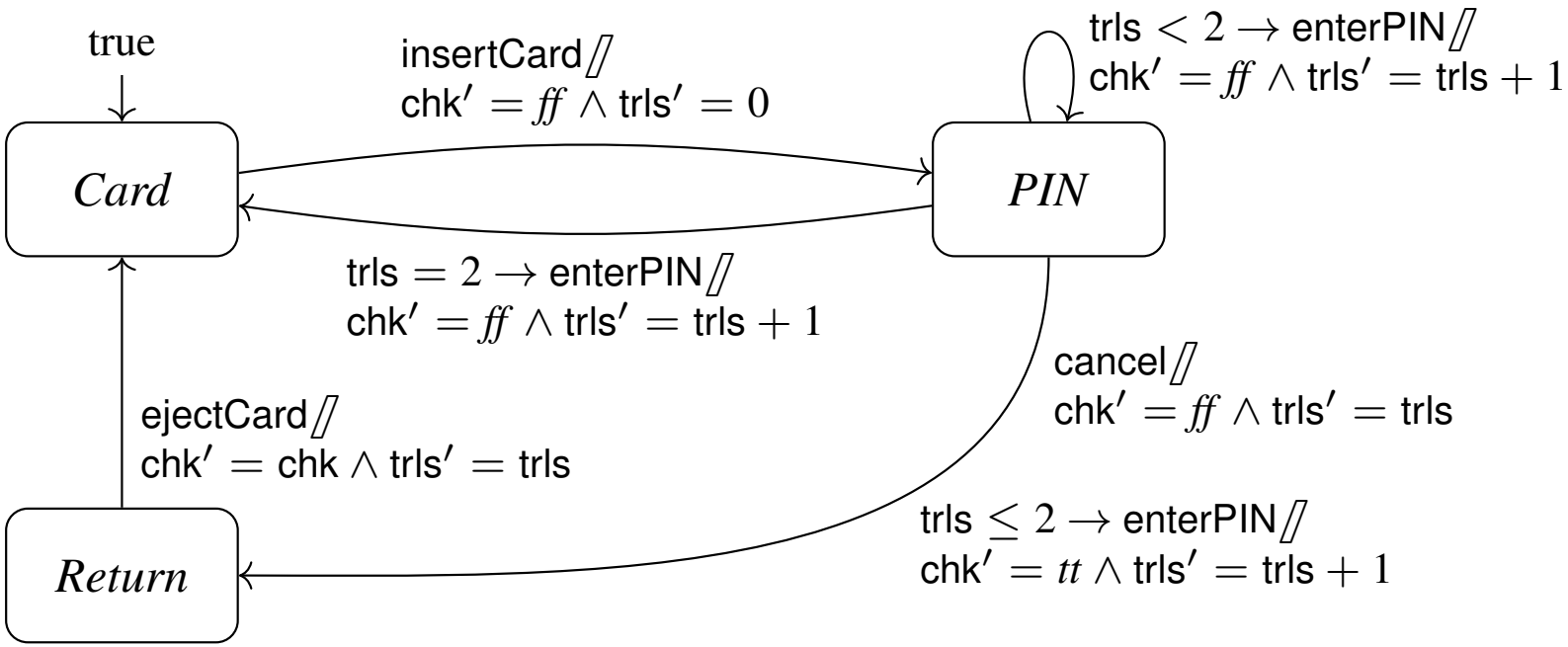
- parallel composition $\kappa_\otimes$
  - binary constructor: $Sp \rightsquigarrow_{\kappa_\otimes} \langle Sp'_1, Sp'_2 \rangle$
  - synchronous product of edts with composable signatures

Operational Event/Data Specifications (1)

More constructive specification style

- graphical representation (like STS, UML protocol state machines)
- can be faithfully expressed in $\mathcal{E}^{\downarrow}$

Example: Specification ATM_2 with $\Sigma_2 = (\{\text{insertCard}, \ldots\}, \{\text{chk}, \text{trls}\})$



Stepwise refinement in $\mathcal{E}^{\downarrow}$: $ATM_0 \rightsquigarrow ATM_1 \overset{?}{\rightsquigarrow} ATM_2$

Operational Event/Data Specifications (2)

Operational ed specification $O = (\Sigma, C, T, (c_0, \varphi_0))$ over ed signature Σ

- control states C
- transition relation specification $T \subseteq C \times \Phi(\Sigma) \times E(\Sigma) \times \Psi(\Sigma) \times C$
  - separate precondition in $\Phi(\Sigma)$ and transition predicate in $\Psi(\Sigma)$
- initial control state $c_0 \in C$, initial state predicate $\varphi_0 \in \Phi(\Sigma)$
  - all control states syntactically reachable from c_0

(Loose) semantics of O given by model class of edts with $M \in \mathrm{Mod}(O)$ if

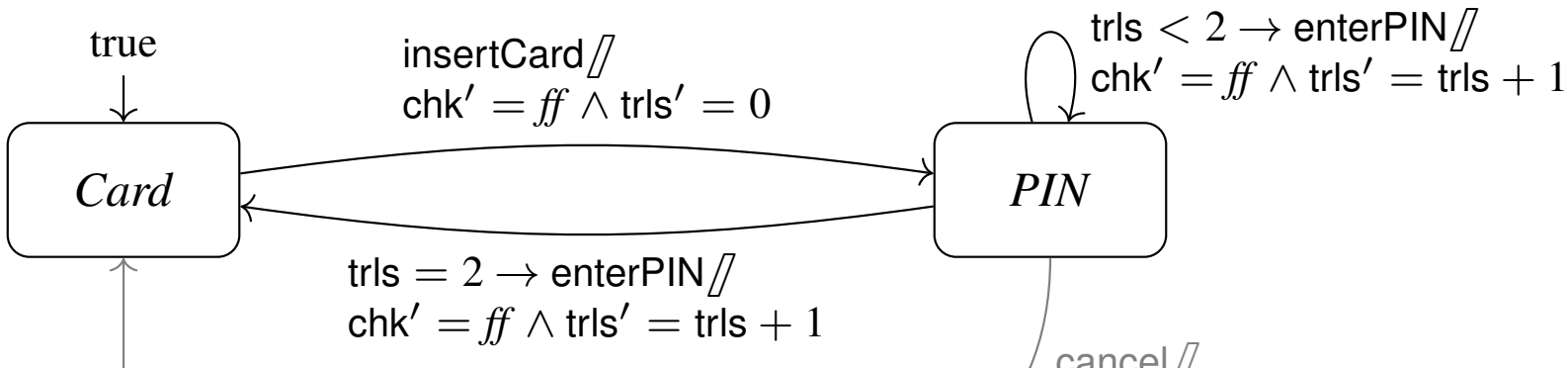
- $R(M)$ only shows transitions allowed by T
  - for all $((c, \omega), (c', \omega')) \in R(M)_e$ there is a $(c, \varphi, e, \psi, c') \in T$ with $\omega \models^{\mathcal{D}}_{A(\Sigma)} \varphi$ and $(\omega, \omega') \models^{\mathcal{D}}_{A(\Sigma)} \psi$
- $R(M)$ realises T for satisfied preconditions
  - for all $(c, \varphi, e, \psi, c') \in T$ and $\omega \models^{\mathcal{D}}_{A(\Sigma)} \varphi$, there is a $((c, \omega), (c', \omega')) \in R(M)_e$ with $(\omega, \omega') \models^{\mathcal{D}}_{A(\Sigma)} \psi$

Expressiveness of $\mathcal{E}^{\downarrow}$

Theorem For every operational ed specification O with finitely many control states there is an ed sentence ϱ_O such that

$$M \in \mathrm{Mod}(O) \iff M \models^{\mathcal{E}^{\downarrow}}_{\Sigma(O)} \varrho_O$$

Example



$$\downarrow Card\,.\,\langle \mathsf{insertCard}/\!/\mathsf{chk}' = ff \wedge \mathsf{trls}' = 0\rangle$$
$$\downarrow PIN\,.\,(@Card)[\mathsf{insertCard}/\!/\mathsf{chk}' = ff \wedge \mathsf{trls}' = 0]PIN \wedge$$
$$[\mathsf{insertCard}/\!/\mathsf{chk}' = tt \vee \mathsf{trls}' \neq 0]\mathrm{false} \wedge$$
$$[\{\mathsf{enterPIN}, \mathsf{cancel}, \mathsf{ejectCard}\}]\mathrm{false} \wedge \ldots$$

ATM-Example: Refinement in $\mathcal{E}^{\downarrow}$ (1)

Refinement chain for ATM specification

$$ATM_0 \rightsquigarrow ATM_1 \rightsquigarrow_{\kappa_\iota} ATM_2 \rightsquigarrow_{\kappa_\otimes;\kappa_\alpha} \langle ATM_3, CC \rangle$$

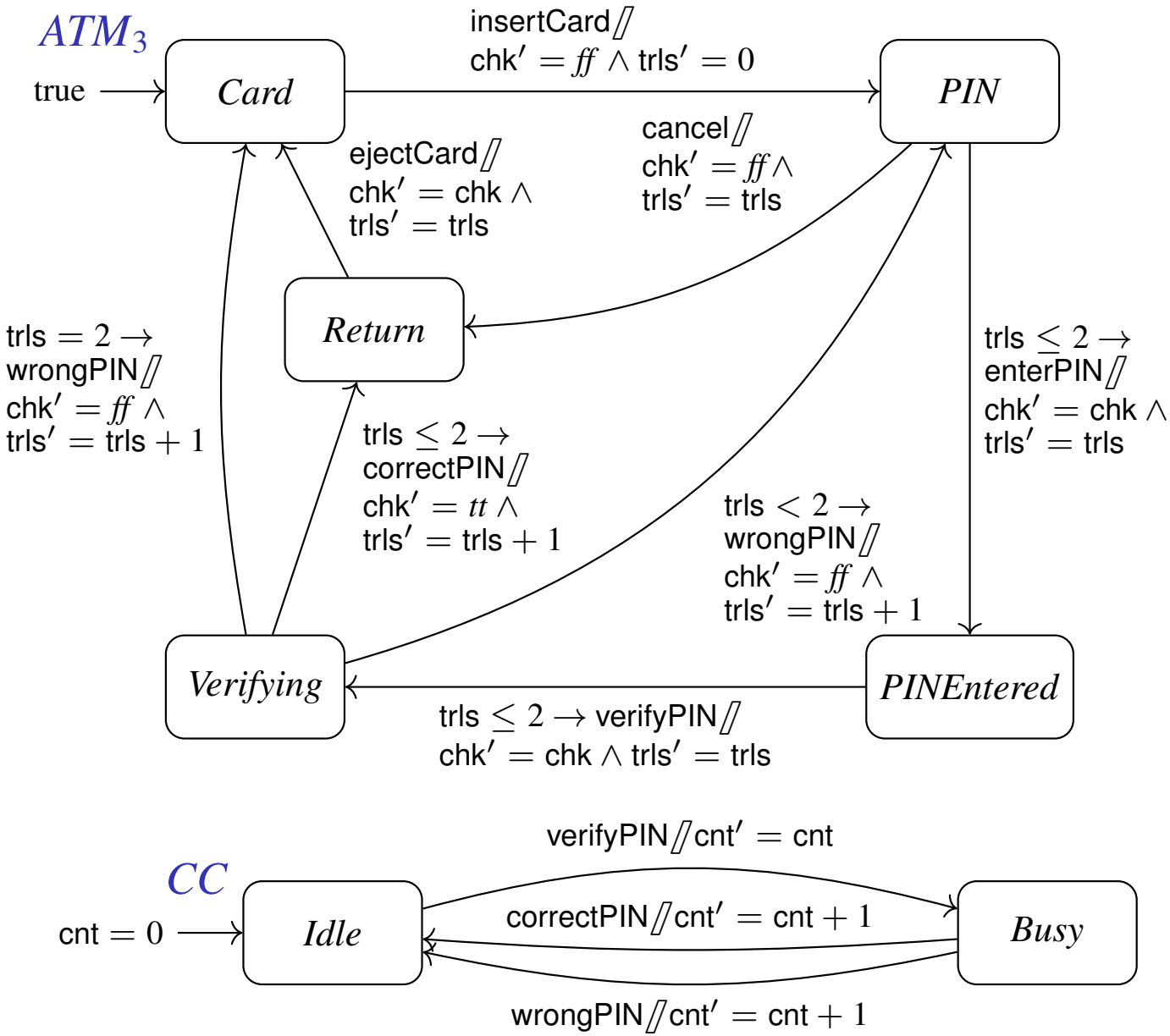
For $ATM_1 \rightsquigarrow_{\kappa_\iota} ATM_2$

- restriction constructor with $\iota : \Sigma_1 \hookrightarrow \Sigma_2$ injective

For $ATM_2 \rightsquigarrow_{\kappa_\otimes;\kappa_\alpha} \langle ATM_3, CC \rangle$

- event refinement constructor κ_α
- parallel composition constructor $\kappa_\otimes$ to two components

ATM-Example: Components



ATM-Example: Refinement in $\mathcal{E}^{\downarrow}$ (2)

Refinement chain for ATM specification

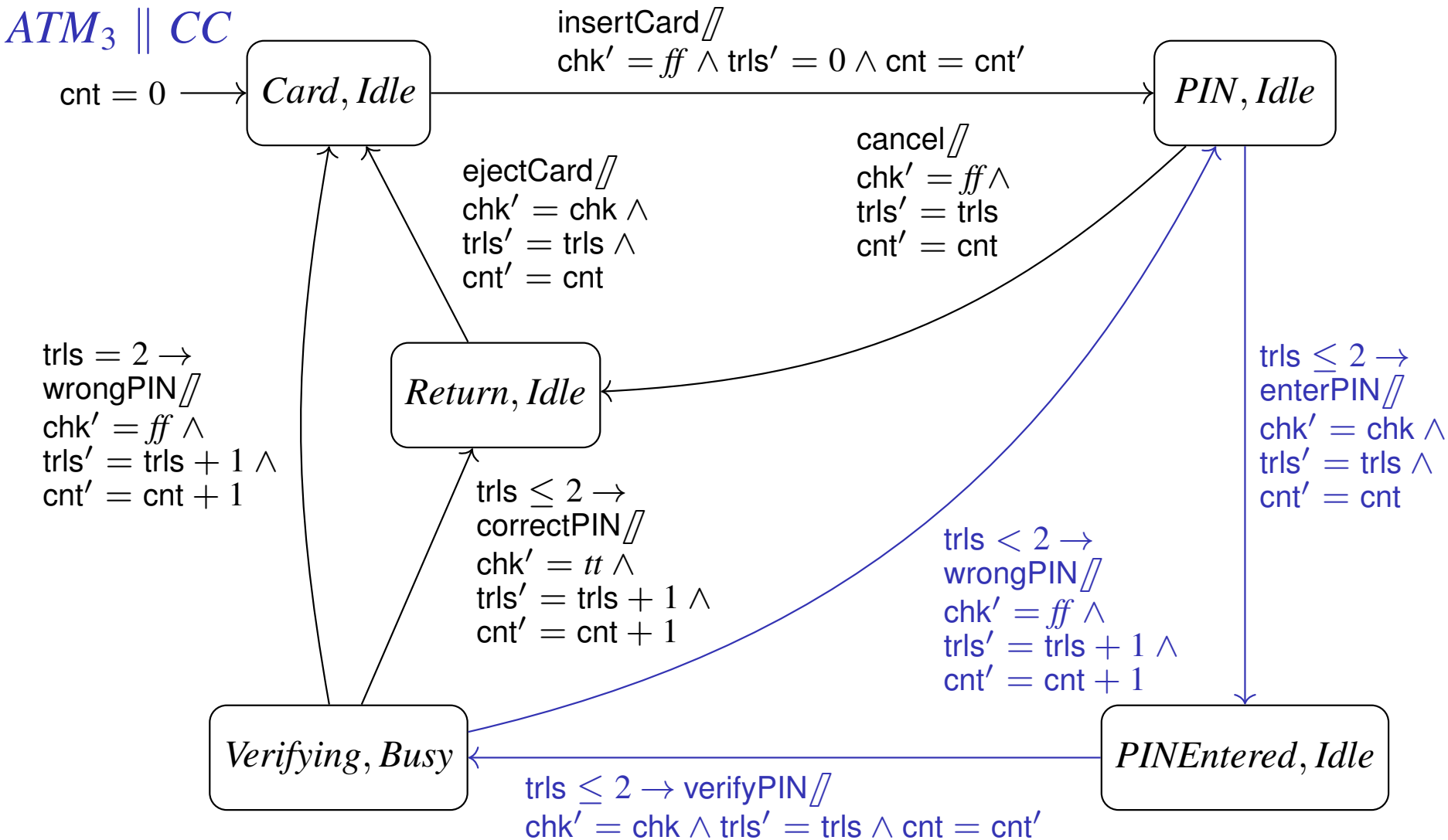
$$ATM_0 \rightsquigarrow ATM_1 \rightsquigarrow_{\kappa_\iota} ATM_2 \rightsquigarrow_{\kappa_\otimes;\kappa_\alpha} \langle ATM_3, CC \rangle$$

Replace $ATM_2 \rightsquigarrow_{\kappa_\otimes;\kappa_\alpha} \langle ATM_3, CC \rangle$ by

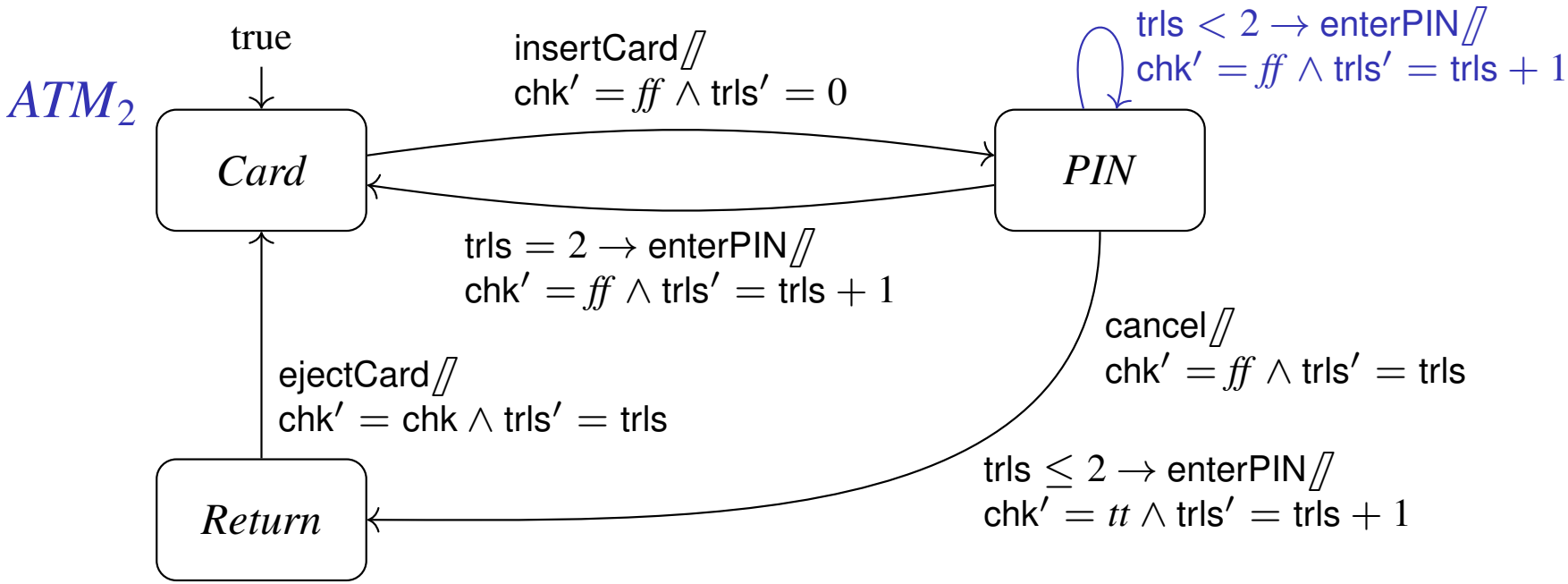
$$ATM_2 \rightsquigarrow_{\kappa_\alpha} ATM_3 \parallel CC \rightsquigarrow_{\kappa_\otimes} \langle ATM_3, CC \rangle$$

- syntactic parallel composition of operational ed specifications

ATM-Example: Syntactic Parallel Composition



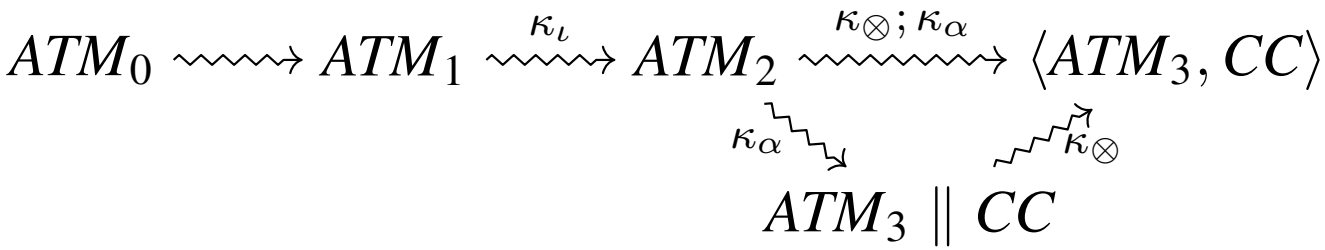
ATM-Example: Event Refinement



$ATM_2 \rightsquigarrow_{\kappa_\alpha} ATM_3 \parallel CC$

- $\{\mathsf{chk}, \mathsf{trls}\} \subseteq \{\mathsf{chk}, \mathsf{trls}, \mathsf{cnt}\}$
- $\alpha(\mathsf{enterPIN}) = (\mathsf{enterPIN}; \mathsf{verifyPIN}; (\mathsf{correctPIN} + \mathsf{wrongPIN}))$

ATM-Example: Refinement in $\mathcal{E}^{\downarrow}$ (3)



Proposition Let O_1, O_2 be operational ed specifications with composable signatures. Then

$$\mathrm{Mod}(O_1) \otimes \mathrm{Mod}(O_2) \subseteq \mathrm{Mod}(O_1 \parallel O_2)$$

(Converse inclusion does not hold.)

Theorem Let Sp be an (axiomatic or operational) ed specification, O_1, O_2 operational ed specifications with composable signatures, and κ a constructor from $\Sigma(O_1) \otimes \Sigma(O_2)$ to $\Sigma(Sp)$. Then

if $Sp \rightsquigarrow_\kappa O_1 \parallel O_2$, then $Sp \rightsquigarrow_{\kappa_\otimes ; \kappa} \langle O_1, O_2 \rangle$

Further Developments

Institutional formulation of $\mathcal{E}^{\downarrow}$

- institution $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$ over an underlying data institution $\mathcal{D}$
- change of data institution (like propositional to first-order logic) as further refinement step

Rolf Hennicker, A. K., Alexandre Madeira. Hybrid Dynamic Logic Institutions for Event/Data-based Systems. Formal Aspect. Comp., 2021

Rolf Hennicker, A. K. Specification of Systems with Parameterised Events: An Institution-independent Approach. J. Log. Alg. Meth. Program., 2022

Encoding of (simple) UML state machines and composite structures

- Parameterised input/output events
- Theoroidal institution comorphism to CASL for theorem proving

Tobias Rosenberger, Saddek Bensalem, A. K., Markus Roggenbach. Institution-based Encoding and Verification of Simple UML State Machines in CASL/SPASS. WADT 2020

Tobias Rosenberger, A. K., Markus Roggenbach. An Institutional Approach to Communicating UML State Machines. FASE 2022

Conclusions and Future Work

Specifying event/data-based systems in $\mathcal{E}^{\downarrow}$

- Expressive logic through combination of dynamic and hybrid features
- Support for both abstract requirements specifications and concrete implementations
- Support for stepwise refinement through constructor implementations

`https://bitbucket.org/knappale/edhl`

- Integrate other formalisms into $\mathcal{E}^{\downarrow}$-development process
  - TLA; similar to operational specifications: Event-B, UML state machines
  - communication compatibility for input/output
- Beyond bisimulation invariance for hybrid-free sentences
- Proof system for $\mathcal{E}^{\downarrow}$, including data states