A Hybrid Dynamic Logic for Event/Data-based Systems

Rolf Hennicker

Ludwig-Maximilians-Universität München

Alexandre Madeira

CIDMA, Universidade de Aveiro & QuantaLab, Universidade do Minho

Alexander Knapp

Universität Augsburg

Specifying Event/Data-based Systems (1)

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, UML behavioural/protocol state machines
- Property-oriented approaches (abstract specification)
  - modal (temporal, dynamic) logics, TLA
- Checking whether a concrete model satisfies certain abstract properties

Specifying Event/Data-based Systems (2)

Goals

- Common logical formalism for specifying event/data-based systems on various levels of abstraction
- Program development by stepwise refinement (“correct by construction”)
  - 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

Example: Specifying an ATM

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

Axiomatic specification using formulæ, like

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

“Whenever either a correct PIN has been entered or the transaction has been cancelled, the card can eventually be ejected.”

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

“Whenever either 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.”

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 $\omega \models^{\mathcal{D}}_{A} \varphi$
- transition predicates $\psi \in \Psi(\Sigma)$ with $(\omega, \omega') \models^{\mathcal{D}}_{A} \psi$

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: $\boldsymbol{E}^*$; enterPIN$/\!\!/$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 $M, \gamma_0, v \models^{\mathcal{E}\downarrow}_{\Sigma} \varrho$ 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 $\Sigma_0 = (\{\text{insertCard}, \ldots\}, \{\text{chk}\})$ and Ax_0:

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

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

$$\downarrow x_0 \,.\, [\boldsymbol{E}^*; (\text{enterPIN} /\!\!/ \text{chk}' = tt) + \text{cancel}]\langle \boldsymbol{E}^*; \text{ejectCard}\rangle x_0$$
$$\langle \text{insertCard} /\!\!/ \text{chk}' = \mathit{ff}\rangle \mathrm{true} \wedge$$
$$[\text{insertCard} /\!\!/ \neg(\text{chk}' = \mathit{ff})]\mathrm{false} \wedge [-\text{insertCard}]\mathrm{false} \;, \quad \ldots$$

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

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

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

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

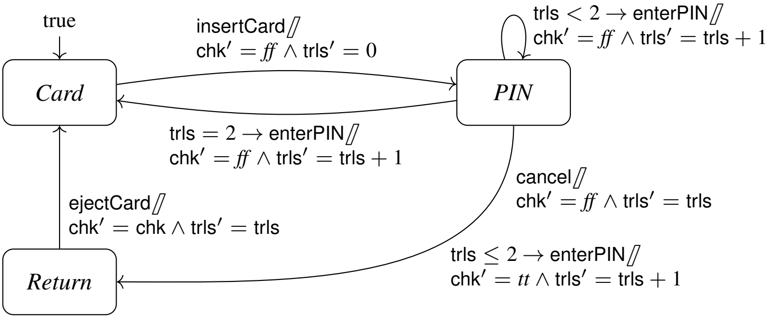
- parallel composition $\kappa_\otimes$
  - binary constructor: $Sp \rightsquigarrow_{\kappa_\otimes} \langle Sp'_1, Sp'_2 \rangle$
  - synchronous product of edts with composable signatures

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 $\Sigma_2 = (\{\mathsf{insertCard}, \ldots\}, \{\mathsf{chk}, \mathsf{trls}\})$



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

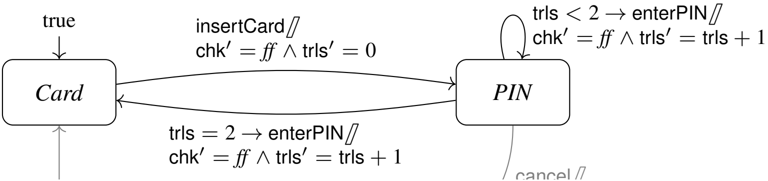
- $R(M)$ only shows transitions allowed by T
  - for all $((c, \omega), (c', \omega')) \in R(M)_e$ there is a $(c, \varphi, e, \psi, c') \in T$ with $\omega \models^{\mathcal{D}}_{A(\Sigma)} \varphi$ and $(\omega, \omega') \models^{\mathcal{D}}_{A(\Sigma)} \psi$
- $R(M)$ realises T for satisfied preconditions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



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

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

Refinement chain for ATM specification

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

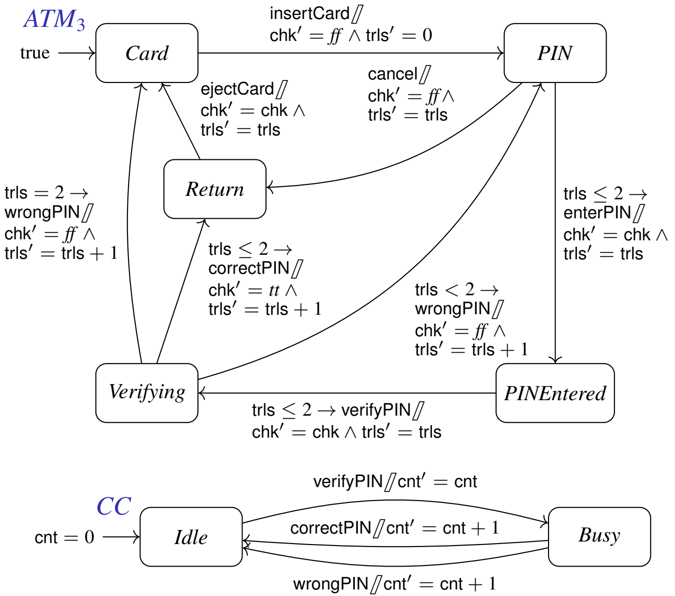
For $ATM_1 \rightsquigarrow_{\kappa_\iota} ATM_2$

- 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

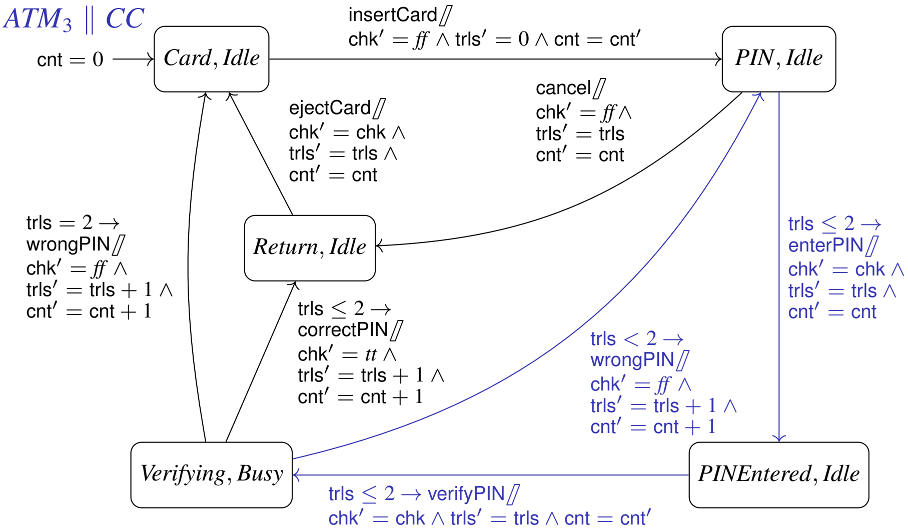
$$ATM_0 \rightsquigarrow ATM_1 \rightsquigarrow_{\kappa_\iota} ATM_2 \rightsquigarrow_{\kappa_\otimes;\kappa_\alpha} \langle ATM_3, CC\rangle$$

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

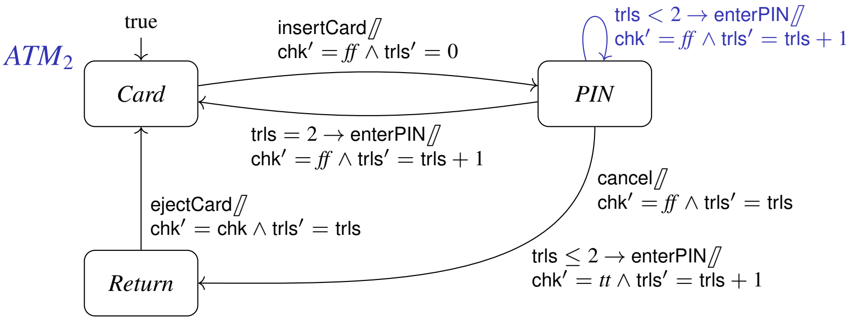
$$ATM_2 \rightsquigarrow_{\kappa_\alpha} ATM_3 \parallel CC \rightsquigarrow_{\kappa_\otimes} \langle ATM_3, CC\rangle$$

- 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)

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

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

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

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

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