Events, permissions, and obligations

. . . and their refinement

Stephan Merz

(with thanks to Frédéric Cuppens and Dominique Méry)

INRIA Lorraine & LORIA, Nancy

INRIA

Loria

Context

- Project on information security
  - access control
  - information flow
- Static system model : who may / must (not) do what ?
  - identify organizations, roles, activities, contexts, etc
  - assign permissions / user rights and responsibilities
- Integration with dynamic system model
  - (temporal) properties of behaviors
  - stepwise refinement preserving “deontic” properties

Framework: event systems

system $Bank$

constants $Client, Loan, maxDebt$

variables $loans, clt, due, rate$

invariant $\wedge\ loans \subseteq Loan$

$$\wedge\ clt \in [loans \rightarrow Client] \wedge due \in [loans \rightarrow \mathbb{N}] \wedge rate \in [loans \rightarrow \mathbb{N}]$$

$$\wedge\ \forall c \in Client : \sum\{due(ll) : ll \in loans \wedge clt(ll) = c\} \leq maxDebt$$

initial $loans = \emptyset$

event $newLoan(c : Client, l : Loan, sum : \mathbb{N}, dur : \mathbb{N}) \equiv$

$$\wedge\ l \notin loans \wedge sum + \sum\{due(ll) : ll \in loans \wedge clt(ll) = c\} \leq maxDebt$$

$$\wedge\ loans' = loans \cup \{l\} \wedge clt' = clt \cup \{l \mapsto c\}$$

$$\wedge\ due' = due \cup \{l \mapsto sum\} \wedge rate' = rate \cup \{l \mapsto sum/dur\}$$

event $payRate(l : Loan) \equiv$

$$\wedge\ l \in loans$$

$$\wedge\ due' = due \oplus \{l \mapsto due(l) - rate(l)\}$$

$$\wedge\ clt' = clt \wedge rate' = rate$$

Properties (safety)

- Stable predicates

$$\frac{P \wedge e(x) \Rightarrow P' \quad \text{for all events } e}{\textbf{stable } P}$$

- Invariants

$$\frac{Init \Rightarrow P \qquad \textbf{stable } P}{\textbf{inv } P} \qquad\qquad \frac{\textbf{inv } P \qquad P \Rightarrow Q}{\textbf{always } Q}$$

- Proof obligation : **inv** Inv

  for the declared system invariant Inv

Adding fairness conditions

- Event systems describe what *can* occur
- Fairness ensures that events *do* occur eventually

$$
\begin{array}{ll}
\textbf{event} & payRate(l : Loan) \equiv \\
 & \wedge\, l \in loans \\
 & \wedge\, due' = due \oplus \{l \mapsto due(l) - rate(l)\} \\
 & \wedge\, clt' = clt \wedge rate' = rate \\
\textbf{fairness} & l \in loans \wedge due(l) > 0
\end{array}
$$

- This talk : weak fairness
  - if condition persists, event must eventually occur
  - condition may be stronger than guard

Properties (liveness)

- $F \leadsto G$: every F will be followed by G

- verification rules

$$\frac{\begin{array}{c} P \wedge a(x) \wedge \neg e(t) \Rightarrow P' \vee Q' \quad \text{for all events } a \\ P \Rightarrow \mathit{fair}_e(t) \end{array}}{P \leadsto Q \vee (P \wedge e(t))}$$

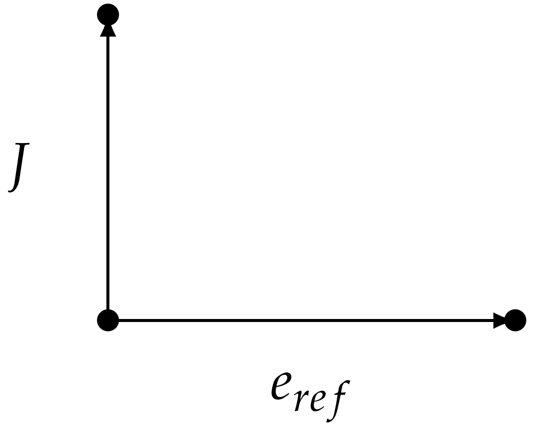
$$\frac{P \wedge e(t) \Rightarrow Q'}{P \wedge e(t) \leadsto Q} \qquad \frac{F \Rightarrow G}{F \leadsto G} \qquad \frac{\mathbf{inv}\ I \qquad I \wedge F \leadsto G \vee \neg I}{F \leadsto G}$$

$$\frac{\forall x \in S : F(x) \leadsto G \vee (\exists y \in S : y \prec x \wedge F(y)) \qquad (S, \prec) \text{ well-founded}}{(\exists x \in S : F(x)) \leadsto G \qquad (x \text{ not free in } G)}$$

$$\frac{F \leadsto G \qquad G \leadsto H}{F \leadsto H} \qquad \frac{F \leadsto H \qquad G \leadsto H}{F \vee G \leadsto H} \qquad \frac{F \leadsto G}{(\exists x : F) \leadsto (\exists x : G)}$$

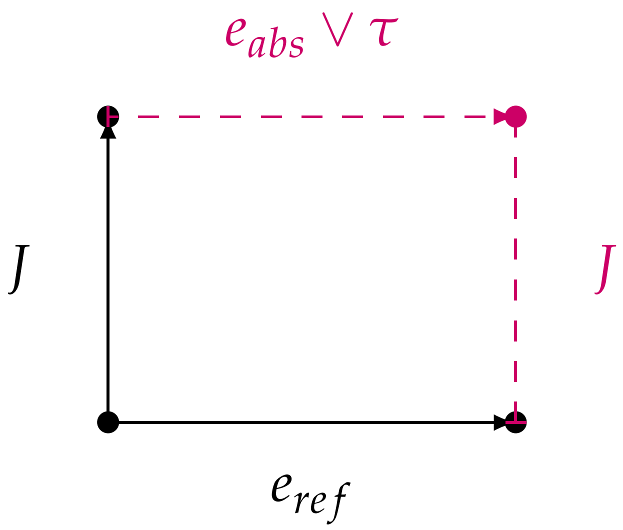
Refinement : intuition

- add detail to model, but preserve properties
  - different data representation, related by linking invariant J
  - refine grain of atomicity of events
- map concrete events to abstract ones (maybe stutter)



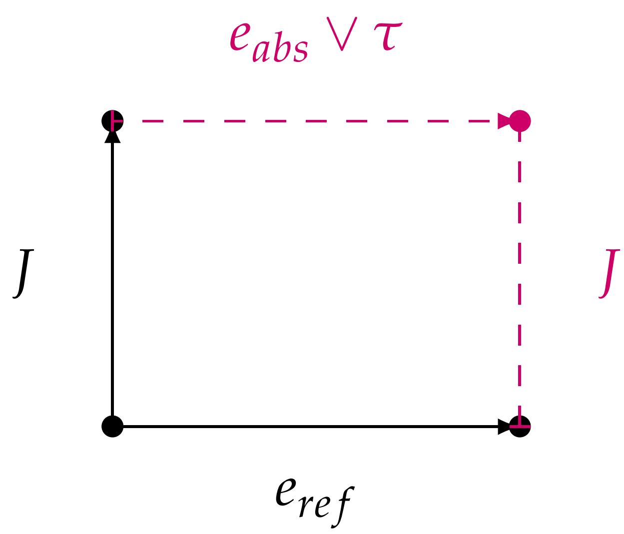
Refinement : intuition

- add detail to model, but preserve properties
  - different data representation, related by linking invariant J
  - refine grain of atomicity of events
- map concrete events to abstract ones (maybe stutter)



Refinement : intuition

- add detail to model, but preserve properties
  - different data representation, related by linking invariant J
  - refine grain of atomicity of events
- map concrete events to abstract ones (maybe stutter)



common extra conditions :

- eventually perform abstract events
- relative deadlock freedom

here : preserve fairness

Refinement : proof obligations

- simulation of initial condition

$$Init_{ref} \Rightarrow \exists var_{abs} : Init_{abs} \wedge J$$

- step simulation (possibly stuttering)

$$er(t) \wedge J \Rightarrow \exists u, var'_{abs} : ea(u) \wedge J' \qquad (er \text{ refines } ea)$$

$$er(t) \wedge J \Rightarrow \exists var'_{abs} : var'_{abs} = var_{abs} \wedge J' \qquad (er \text{ new event})$$

- refinement of fairness constraints $\quad (er_1, \ldots, er_n \text{ refine } ea)$

$$\begin{aligned} \textbf{true} \leadsto\ & \vee \neg(\exists var_{abs} : fair_{ea}(u) \wedge J) \\ & \vee (\exists t_1 : er_1(t_1)) \vee \ldots \vee (\exists t_n : er_n(t_n)) \end{aligned}$$

Refinement : properties

- simulation of traces

  for every trace of the concrete system Ref

  there is a corresponding trace of the abstract system Abs

Refinement : properties

- simulation of traces

  for every trace of the concrete system Ref

  there is a corresponding trace of the abstract system Abs

- preservation of properties modulo linking invariant

$$
\begin{array}{lcl}
Abs \models \textbf{stable}\ P & \Rightarrow & Ref \models \textbf{stable}\ (\exists var_{abs} : P \land J) \\
Abs \models \textbf{inv}\ P & \Rightarrow & Ref \models \textbf{inv}\ \bar{P} \\
Abs \models \textbf{always}\ P & \Rightarrow & Ref \models \textbf{always}\ \bar{P} \\
Abs \models F \leadsto G & \Rightarrow & Ref \models \bar{F} \leadsto \bar{G}
\end{array}
$$

Permissions & obligations

- Who may/must do what, under what circumstances ?
  - static model of entities and activities (Or-BAC)
    represented as constants and events
  - specify permissions / rights and obligations
    add corresponding predicates to event definitions

Permissions & obligations

- Who may/must do what, under what circumstances ?
  - static model of entities and activities (Or-BAC)
    represented as constants and events
  - specify permissions / rights and obligations
    add corresponding predicates to event definitions
- Relation with system model ?
  - verify "deontic" properties of model
  - and adapt refinement relation

Representing permissions

- Extend description of events

  event $newLoan(c : Client, l : Loan, sum : \mathbb{N}, dur : \mathbb{N}) \equiv \ldots$

  permission $l \notin loans \wedge risk(c, sum) \in \{low, medium\}$

  interdiction $risk(c, sum) = high$

- Verification conditions ensure that annotations hold
  - invariant and permission implies guard
  - invariant and interdiction implies negation of guard

    always $\neg(e(t) \wedge intd_e(t))$

Representing obligations

- Similarly add obligation predicates

  event $payRate(l : Loan) \equiv \ldots$

  obligation $l \in loans \wedge due(l) > 0$

- Temporal interpretation

  strict obligation $\quad obl_e(t) \rightsquigarrow e(t)$

  weak obligation $\quad obl_e(t) \rightsquigarrow \neg obl_e(t) \vee e(t)$ $\qquad$ [this is just weak fairness!]

- We know how to establish these properties

Refinement : preserving properties

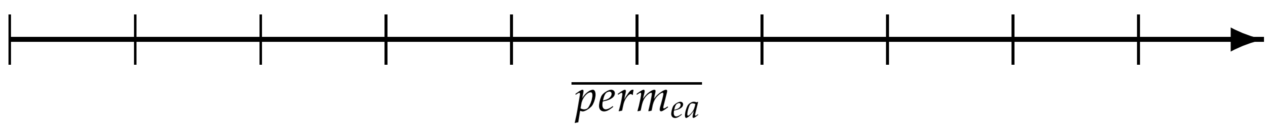
- Obligations & interdictions : nothing to prove
  - expressed as (linear-time) properties of traces
  - hence preserved by refinement

Refinement : preserving properties

- Obligations & interdictions : nothing to prove
  - expressed as (linear-time) properties of traces
  - hence preserved by refinement
- Permissions : more problematic
  - refinement does not preserve branching behavior
  - what should be preserved across non-atomic refinement ??
  - refined event won't be executable whenever abstract one is

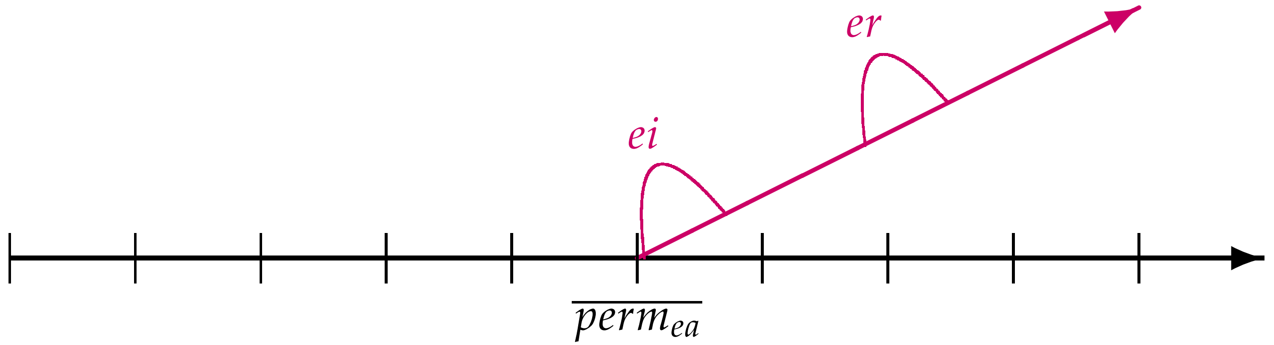
Refinement of permissions

- Idea : refine abstract-level permission
  - by a concrete-level permission (to start a branch)
  - *and* a concrete-level obligation (to simulate the event)



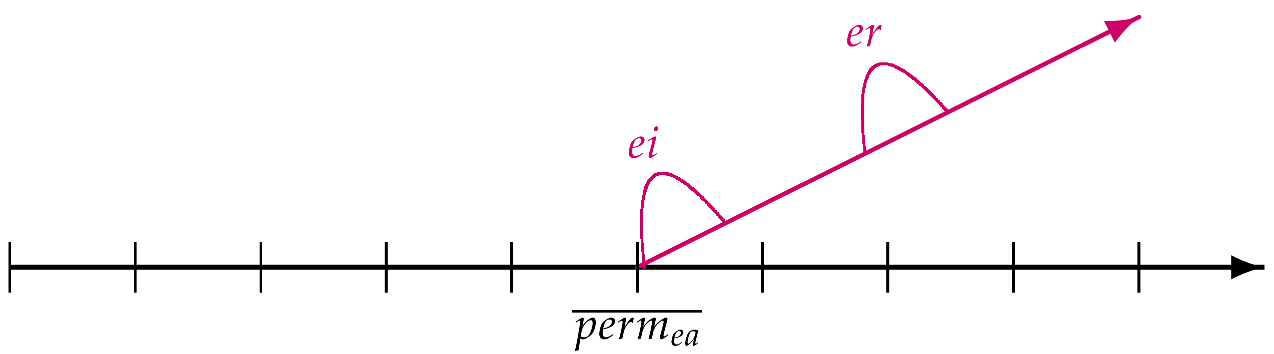
Refinement of permissions

- Idea : refine abstract-level permission
  - by a concrete-level permission (to start a branch)
  - *and* a concrete-level obligation (to simulate the event)



Refinement of permissions

- Idea : refine abstract-level permission
  - by a concrete-level permission (to start a branch)
  - *and* a concrete-level obligation (to simulate the event)



- Formalization (assume ea refined by $er_1, \ldots, er_n$)

  identify "initial events" $ei_1, \ldots, ei_m$ of refined model where

  $\overline{perm_{ea}} \Rightarrow perm_{ei_1} \vee \ldots \vee perm_{ei_m}$ and

  $ei_j \rightsquigarrow \neg\overline{perm_{ea}} \vee er_1 \vee \ldots \vee er_n$

Example

- Refining event *newLoan*

$$\textbf{event}\ \ askLoan(c : Client, l : Loan, sum : \mathbb{N}, dur : \mathbb{N}) \equiv \ldots$$

$$\textbf{permission}\ \ l \notin loans$$

$$\textbf{event}\ \ approveLoan(l : Loan, e : Employee) \equiv \ldots$$

$$\begin{array}{l} \textbf{permission} \\ \qquad \wedge\ l \in non\_approved \\ \qquad \wedge\ \vee\ risk(clt(l), due(l)) = low \wedge rank(e) \geq Clerk \\ \qquad \quad\ \ \vee\ risk(clt(l), due(l)) = medium \wedge rank(e) \geq Manager \end{array}$$

$$\textbf{interdiction}\ \ risk(clt(l), due(l)) = high$$

$$\textbf{obligation}\ \ l \in non\_approved \wedge risk(clt(l), due(l)) \in \{low, medium\}$$

Observations

- Refinement of permissions is transitive
  - introduce explicit permission on “initial” event
  - has to be taken into account when refining further
- Weak interpretation of obligations adequate
  - consider client applying for two loans concurrently
  - no obligation to approve them both

Summing up

- slight extension of event systems
- represent permissions, interdictions, obligations
- property-preserving refinement rules
  - non-atomic refinement of events
  - inheritance of linear-time properties
  - basic branching-time properties : enabledness + liveness
- future work : controllers for security policies