Semantics and Controllability of Time-Aware Business Processes

E. De Angelis [1], F. Fioravanti [1], M.C. Meo [1]
A. Pettorossi [2], M. Proietti [3]

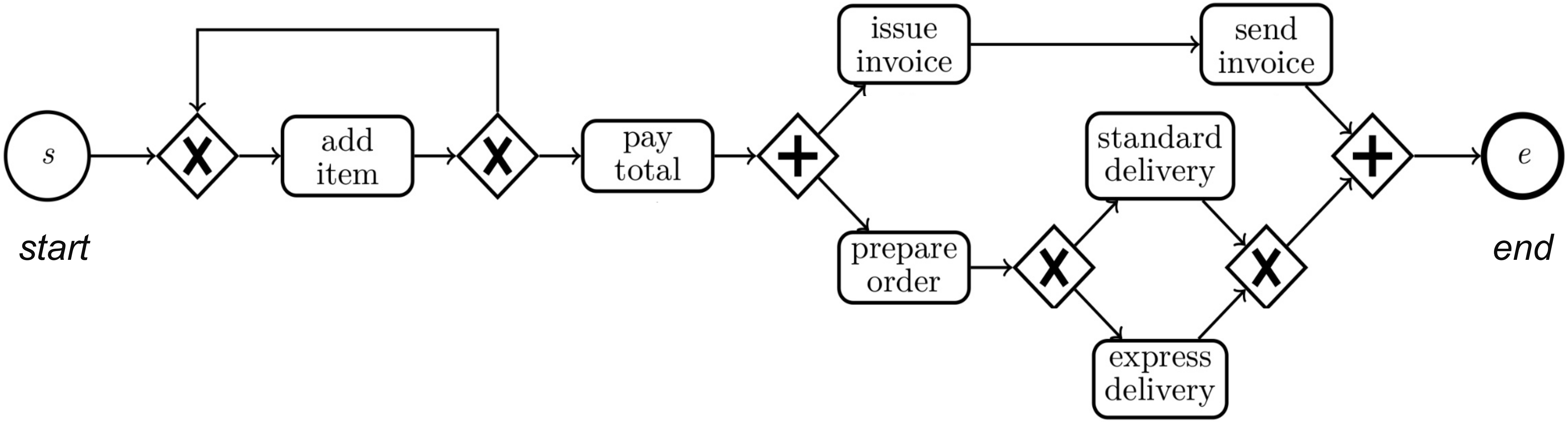
(1) DEC, University “G. d’Annunzio” of Chieti-Pescara, Italy
(2) DICII, University of Rome Tor Vergata, Rome, Italy
(3) CNR-IASI, Rome, Italy

-- in memory of Professor Helena Rasiowa --

Concurrency Specification & Programming 2017
25-27 September 2017, Warsaw (Poland)

Business Processes

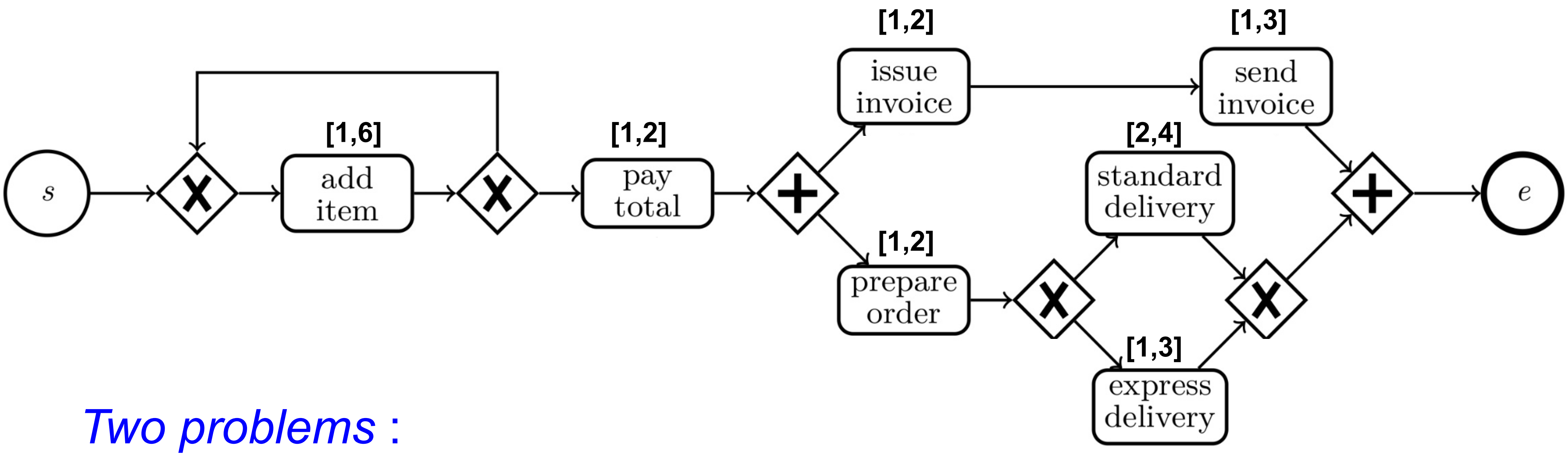
- *Business processes* are 'graphs' for coordinating the activities of an organization towards a business goal.
- *An example: Purchase Order*. A customer adds items to the shopping cart and pays. Then, the vendor issues and sends the invoice, and in parallel, prepares and delivers the order.



There is no information on the durations of tasks.

Time-Aware Business Processes

- *Information on the duration:* Intervals : $d \in [dmin, dmax] \subset \mathbf{N}$

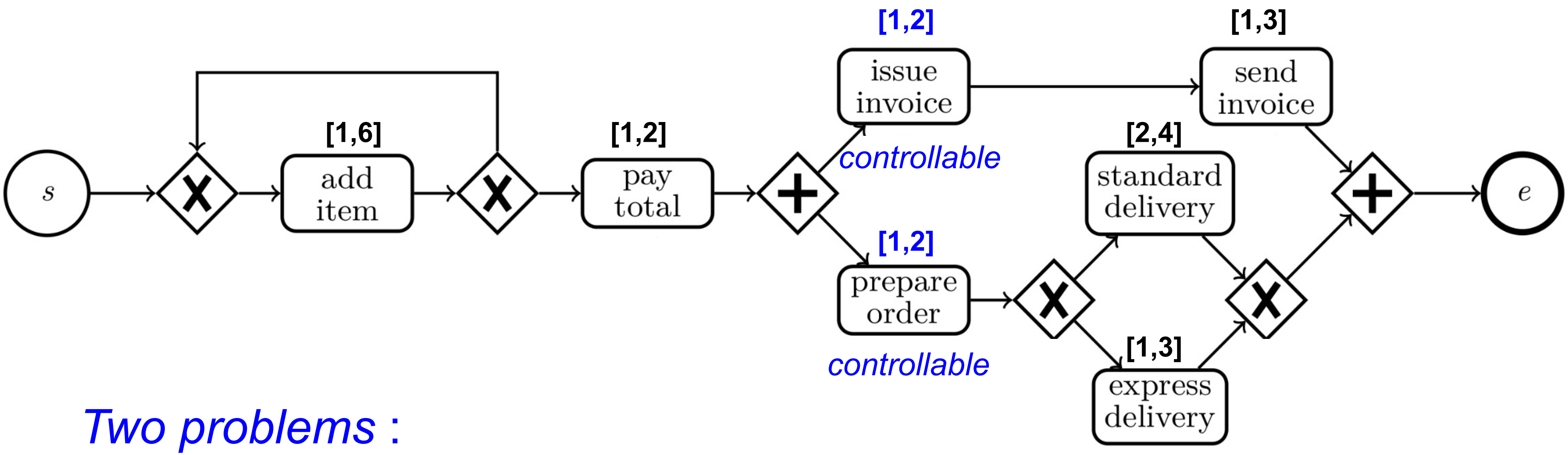


Two problems :

- *Time-Reachability*: checking whether or not to go from s to e takes less than k units of time.

Time-Aware Business Processes

- *Information on the duration:* Intervals : $d \in [dmin, dmax] \subset \mathbf{N}$



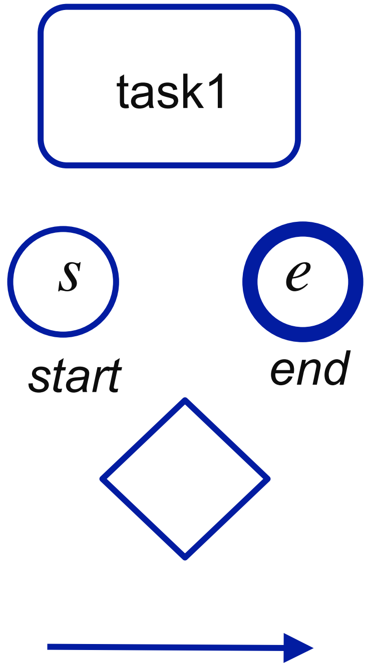
Two problems :

- *Time-Reachability*: checking whether or not to go from s to e takes less than k units of time.
- *Controllability*: finding the durations of some *controllable* tasks so that a given time-reachability property holds.

Business Process Modeling and Notation (BPMN)

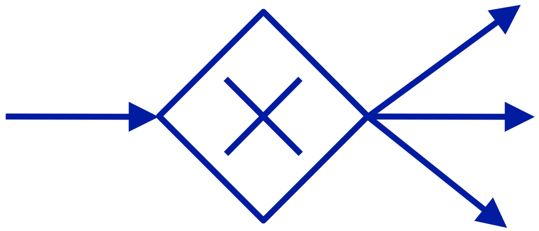
Graphical notation for modeling organizational processes.
BPMN is a standard.

- *Tasks* : atomic activities
- *Events* : something that happens
- *Gateways*: either branching or merging
- *Flows* : order of execution (drawn as *arrows*)

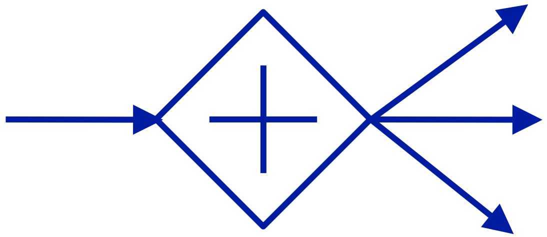


Branch Gateways

- single incoming flow, multiple outgoing flows
- **exclusive** branch gateway (XOR)
  - upon activation of the incoming flow *exactly one* outgoing flow is activated

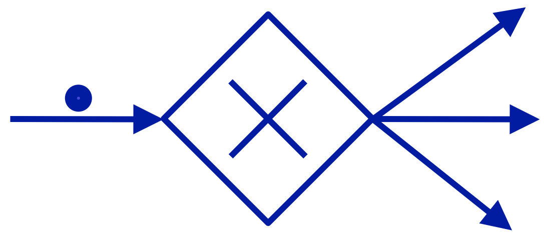


- **parallel** branch gateway (AND)
  - upon activation of the incoming flow *all* outgoing flows are activated

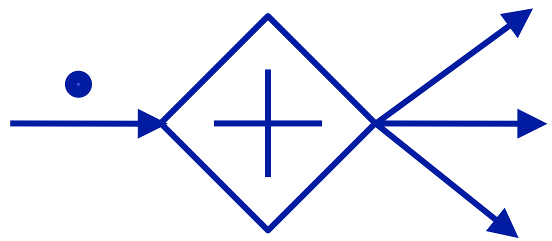


Branch Gateways

- single incoming flow, multiple outgoing flows
- **exclusive** branch gateway (XOR)
  - upon activation of the incoming flow *exactly one* outgoing flow is activated

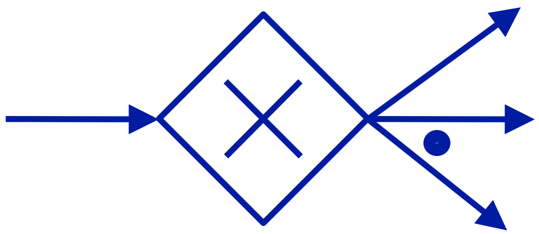


- **parallel** branch gateway (AND)
  - upon activation of the incoming flow *all* outgoing flows are activated

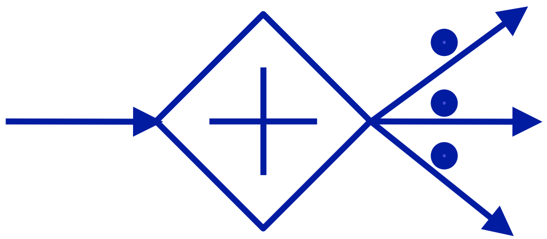


Branch Gateways

- single incoming flow, multiple outgoing flows
- **exclusive** branch gateway (XOR)
  - upon activation of the incoming flow *exactly one* outgoing flow is activated

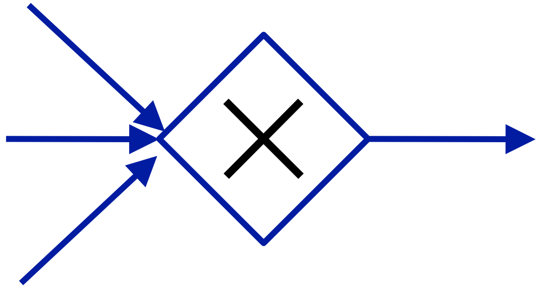


- **parallel** branch gateway (AND)
  - upon activation of the incoming flow *all* outgoing flows are activated

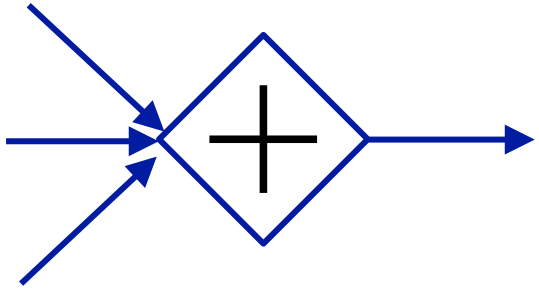


Merge Gateways

- multiple incoming flows, single outgoing flow
- **exclusive** merge gateway (XOR)
  - the outgoing flow is activated upon activation of *one* of the incoming flows

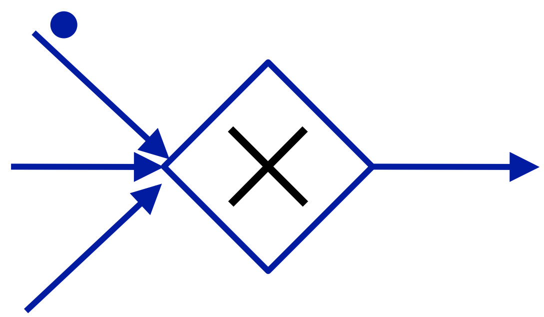


- **parallel** merge gateway (AND)
  - the outgoing flow is activated upon activation of *all* the incoming flows

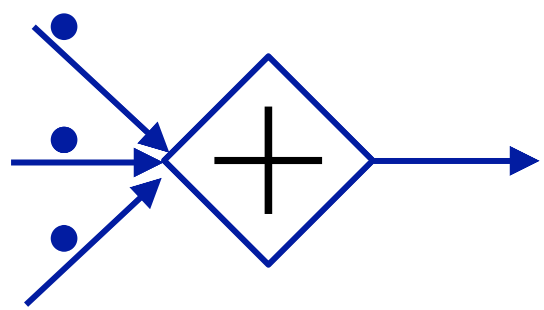


Merge Gateways

- multiple incoming flows, single outgoing flow
- **exclusive** merge gateway (XOR)
  - the outgoing flow is activated upon activation of *one* of the incoming flows

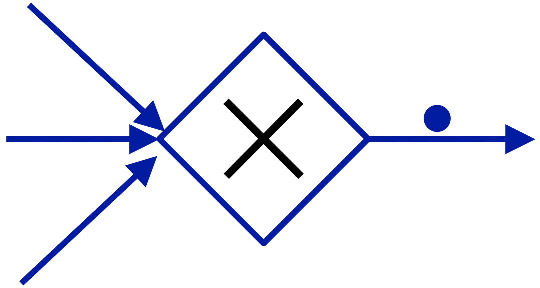


- **parallel** merge gateway (AND)
  - the outgoing flow is activated upon activation of *all* the incoming flows

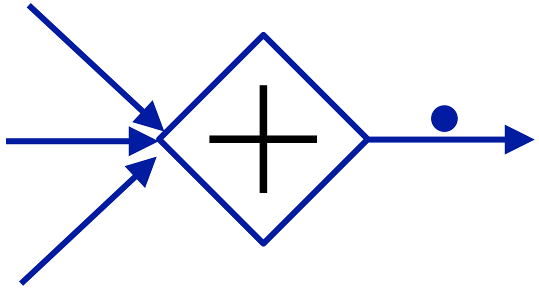


Merge Gateways

- multiple incoming flows, single outgoing flow
- **exclusive** merge gateway (XOR)
  - the outgoing flow is activated upon activation of *one* of the incoming flows



- **parallel** merge gateway (AND)
  - the outgoing flow is activated upon activation of *all* the incoming flows



Semantics of time-aware BPMN

- Transition relation between states: $<F,t> \rightarrow <F',t'>$
- F : a set of *fluents* (i.e., a set of properties that hold at time point t)
  - *begins*(x) — x begins its execution (enactment)
  - *enacting*(x,r) — x is executing with r residual time to completion
  - *completes*(x) — x completes its execution
  - *enables*(x,y) — x enables its successor y

  x, y denote either tasks, or events, or gateways
- *seq*(x,y) — there is an arrow from x to y

- t : time point (i.e., a non-negative integer)

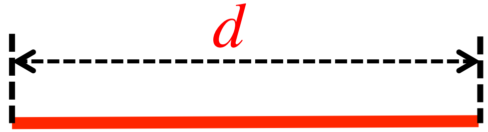
  duration(x,d) — the duration of x is d

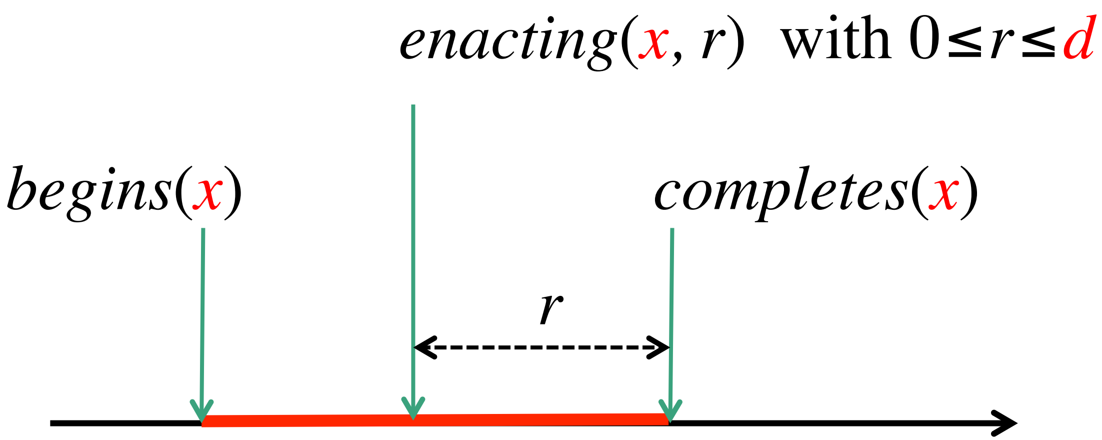
Semantics of time-aware BPMN

$task(x) \leftarrow$

$duration(x, d) \leftarrow 3 \leq d \leq 4$

x is





- durations of events and gateways are assumed to be 0

Semantics of time-aware BPMN

Instantaneous transition: $< F,t > \rightarrow < F',t >$

$$begins(x) \longrightarrow enacting(x,d)$$

$$(S_1) \quad \frac{begins(x) \in F \qquad duration(x,d)}{\langle F, t \rangle \longrightarrow \langle (F \setminus \{begins(x)\}) \cup \{enacting(x,d)\},\ t \rangle}$$

Semantics of time-aware BPMN

Instantaneous transition:

$< F,t > \rightarrow < F',t >$

$$enacting(x,0) \longrightarrow completes(x)$$

$$(S_6) \quad \frac{enacting(x,0) \in F}{\langle F, t\rangle \longrightarrow \langle (F \setminus \{enacting(x,0)\}) \cup \{completes(x)\},\ t\rangle}$$

Semantics of time-aware BPMN

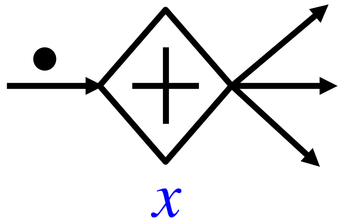
Instantaneous transitions:

$< F,t > \rightarrow\ < F',t >$

$$(S_2) \quad \frac{completes(x) \in F \qquad par\_branch(x)}{\langle F, t\rangle \longrightarrow \langle (F \setminus \{completes(x)\}) \cup \{enables(x,s) \mid seq(x,s)\},\ t\rangle}$$

$$(S_3) \quad \frac{completes(x) \in F \qquad not\_par\_branch(x) \qquad seq(x,s)}{\langle F, t\rangle \longrightarrow \langle (F \setminus \{completes(x)\}) \cup \{enables(x,s)\},\ t\rangle}$$

(S_2) If the parallel branch x completes, then x enables istantaneously all successors s of x



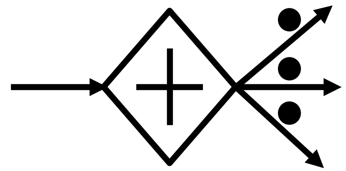
Semantics of time-aware BPMN

Instantaneous transitions: $< F,t > \rightarrow\ < F',t >$

$$(S_2)\quad \frac{completes(x) \in F \qquad par\_branch(x)}{\langle F, t\rangle \longrightarrow \langle (F \setminus \{completes(x)\}) \cup \{enables(x,s) \mid seq(x,s)\},\ t\rangle}$$

$$(S_3)\quad \frac{completes(x) \in F \qquad not\_par\_branch(x) \qquad seq(x,s)}{\langle F, t\rangle \longrightarrow \langle (F \setminus \{completes(x)\}) \cup \{enables(x,s)\},\ t\rangle}$$

(S_2) If the parallel branch x completes,
then x enables istantaneously all successors s of x



x

Semantics of time-aware BPMN

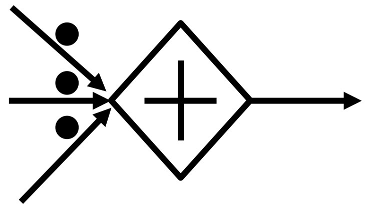
Instantaneous transitions:

$< F,t > \rightarrow\ < F',t >$

$$(S_4)\quad \frac{\forall p\ seq(p,x) \rightarrow enables(p,x) \in F \qquad par\_merge(x)}{\langle F, t\rangle \longrightarrow \langle (F \setminus \{enables(p,x) \mid enables(p,x) \in F\}) \cup \{begins(x)\},\ t\rangle}$$

$$(S_5)\quad \frac{enables(p,x) \in F \qquad not\_par\_merge(x)}{\langle F, t\rangle \longrightarrow \langle (F \setminus \{enables(p,x)\}) \cup \{begins(x)\},\ t\rangle}$$

(S_4) If all predecessors p of the parallel merge x enable x, then the execution of x begins istantaneously.



Semantics of time-aware BPMN

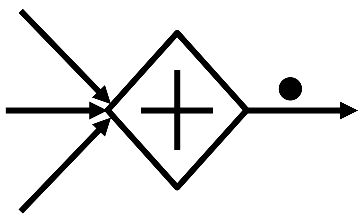
Instantaneous transitions:

$< F,t > \rightarrow \ < F',t >$

$$(S_4) \quad \frac{\forall p \ seq(p,x) \rightarrow enables(p,x) \in F \qquad par\_merge(x)}{\langle F, t\rangle \longrightarrow \langle (F \setminus \{enables(p,x) \mid enables(p,x) \in F\}) \cup \{begins(x)\},\ t\rangle}$$

$$(S_5) \quad \frac{enables(p,x) \in F \qquad not\_par\_merge(x)}{\langle F, t\rangle \longrightarrow \langle (F \setminus \{enables(p,x)\}) \cup \{begins(x)\},\ t\rangle}$$

(S_4) If all predecessors p of the parallel merge x enable x, then the execution of x begins istantaneously.



Semantics of time-aware BPMN

The time-elapsing transition: $< F,t > \rightarrow\ < F',t' >$

$$(S_7)\quad \frac{no\_other\_premises(F) \qquad \exists x\,\exists r\ enacting(x,r) \in F \qquad m > 0}{\langle F, t\rangle \longrightarrow \langle F \ominus m \setminus Enbls,\ t+m\rangle}$$

where: (i) $no\_other\_premises(F)$ holds iff none of the premises of rules S_1–S_6 holds, (ii) $m = min\{r \mid enacting(x,r) \in F\}$, (iii) $F \ominus m$ is the set F of fluents where every $enacting(x,r)$ is replaced by $enacting(x, r-m)$, and (iv) $Enbls = \{enables(p,s) \mid enables(p,s) \in F\}$.

Time elapses when no istantaneous transition can occur.

All enacting tasks proceed in parallel for a time equal to the minimum of all residual times.

Semantics of time-aware BPMN

$< F,t > \rightarrow\ < F',t' >$

at time: T

$$F = \{\ enacting(a,5),\ enacting(b,3)\ \}$$

$\longrightarrow$

at time: $T+3$

$$F = \{\ enacting(a,2),\ enacting(b,0)\ \}$$

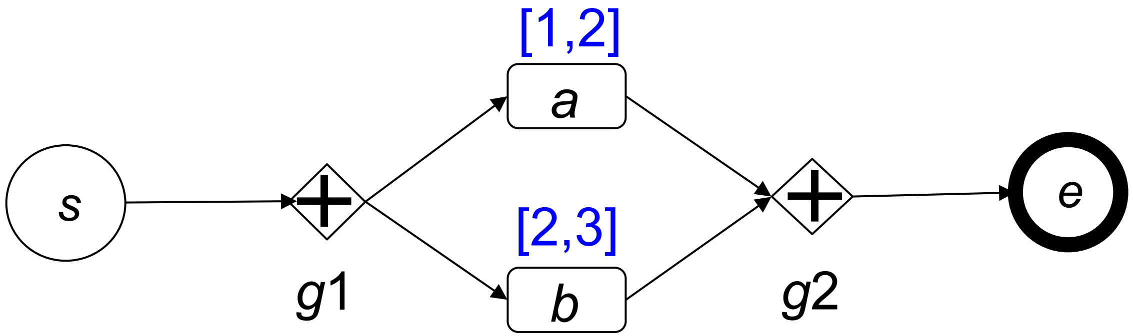
Reachability

State $\langle F,t \rangle$ is reachable iff

for some durations in the given intervals,

$$\langle \{begins(start)\},0 \rangle \rightarrow^* \langle F,t \rangle$$

Semantics in action



<{*begins*(*start*)},0> $\rightarrow^*$ <{*begins*(*g*1)}, 0>

(S_1) $\rightarrow$ <{*enacting*(*g*1,0)}, 0> % *duration*(*g*1,0)

(S_6) $\rightarrow$ <{*completes*(*g*1)}, 0>

(S_2) $\rightarrow$ <{*enables*(*g*1,a),*enables*(*g*1,*b*)}, 0>

(S_5) $\rightarrow$ <{*begins*(*a*),*enables*(*g*1,*b*)}, 0>

(S_1) $\rightarrow$ <{*enacting*(*a*,2),*enables*(*g*1,*b*)}, 0> % 2 in [1,2] for *a*

(S_5 S_1) $\rightarrow^2$ <{*enacting*(*a*,2),*enacting*(*b*,2)}, 0> % 2 in [2,3] for *b*

(S_7) $\rightarrow$ <{*enacting*(*a*,0),*enacting*(*b*,0)}, 2>

(S_6 S_6) $\rightarrow^2$ <{*completes*(*a*),*completes*(*b*)}, 2>

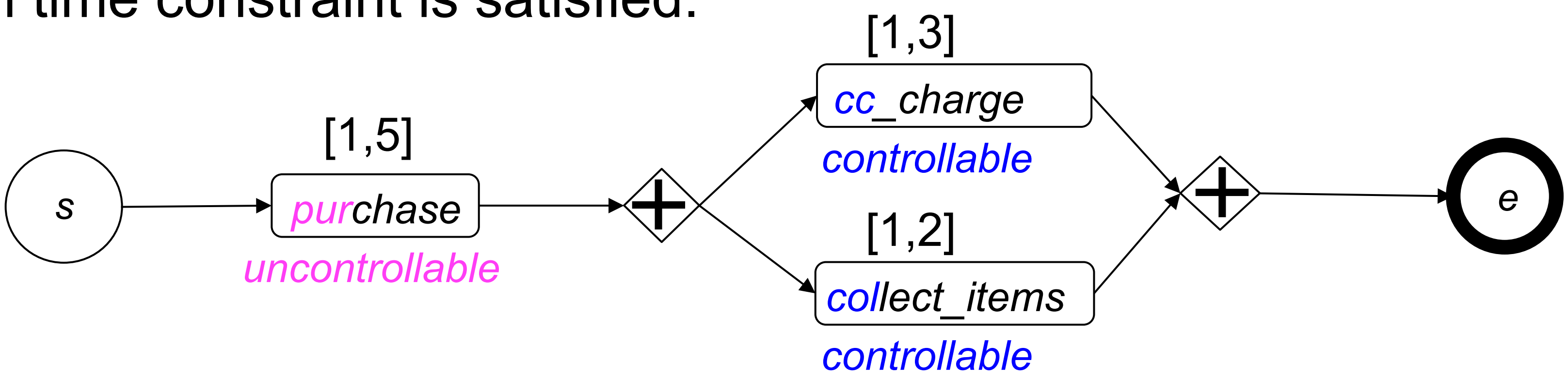
(S_3 S_3) $\rightarrow^2$ <{*enables*(*a*,*g*2),*enables*(*b*,*g*2)}, 2>

(S_4) $\rightarrow$ <{*begins*(*g*2)}, 2>

$\rightarrow^*$ <{*completes*(*end*)}, 2>

Weak Controllability

- Assume:
  - some tasks are *controllable* (e.g., internal to the organization)
  - some tasks are *uncontrollable* (e.g., external to the organization)
- Weak Controllabilty: *For all durations of the uncontrollable tasks* (within the given time intervals), we can *determine durations of the controllable tasks* (within the given time intervals), s.t. a state can be reached and a given time constraint is satisfied.



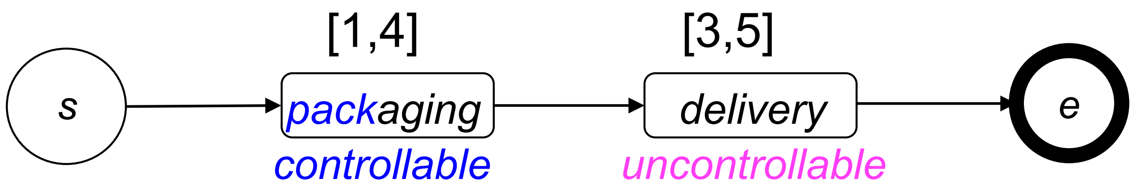
constraint: $3 \leq T_{total} \leq 7$

a solution: *if* $D_{pur}=1$ *then* $D_{cc}=D_{col}=2$ *else* $D_{cc}=D_{col}=1$

Strong Controllability

Weak Controllabilty may not be useful when some uncontrollable tasks occur *after* controllable ones.

- Strong Controllability: We can *determine durations of the controllable tasks* (within the given time intervals) s.t., *for all durations of the uncontrollable tasks* (within the given time intervals), a state can be reached and a given time constraint is satisfied.
- The exact duration of the delivery is not known when packaging.



constraint: $4 \leq T_{total} \leq 7$

a solution: $1 \leq D_{pack} \leq 2$

Solving Controllability Problems using **Constrained Horn Clauses (CHCs)**

- Constrained Horn Clauses: $H \leftarrow \boldsymbol{c}, A_1, \ldots, A_n$
- Use Constrained Horn Clauses to:

(1) Encode the *semantics* of time-aware business processes

(2) Encode *reachability* and *controllability* properties.

---

(3) *Transformation of CHCs*

(4) Applying algorithms for controllability by *using CHC solvers,* (i.e., tools for *Satisfiability Modulo Theory* specialized to CHCs over integers).

(1) Encoding of the semantics

Instantaneous transition: $< F,t > \rightarrow\ < F',t >$

$$begins(x) \longrightarrow enacting(x,d)$$

(1) Encoding of the semantics

Instantaneous transition: $< F,t > \rightarrow < F',t >$

$$begins(x) \longrightarrow enacting(x,d)$$

$$(S_1) \quad \frac{begins(x) \in F \qquad\qquad duration(x,d)}{\langle F, t\rangle \longrightarrow \langle (F \setminus \{begins(x)\}) \cup \{enacting(x,d)\},\ t\rangle}$$

(1) Encoding of the semantics

Instantaneous transition: $< F,t > \rightarrow < F',t >$

$$begins(x) \longrightarrow enacting(x,d)$$

$$(S_1) \quad \frac{begins(x) \in F \qquad\qquad duration(x,d)}{\langle F,t\rangle \longrightarrow \langle (F \setminus \{begins(x)\}) \cup \{enacting(x,d)\},\ t\rangle}$$

$$\boxed{\begin{array}{l} C1.\ tr(s(F,T), s(FU,T), U, C) \leftarrow select(\{begins(X)\}, F),\ \ task\_duration(X,D,U,C), \\ \qquad update(F, \{begins(X)\}, \{enacting(X,D)\}, FU) \end{array}}$$

where U,C are tuples of uncontrollable and controllable durations, resp.

CHC intepreter of time-aware BPMN

$$C1.\ tr(s(F,T), s(FU,T), U, C) \leftarrow select(\{begins(X)\}, F),\ task\_duration(X, D, U, C),\ update(F, \{begins(X)\}, \{enacting(X, D)\}, FU)$$

$$C2.\ tr(s(F,T), s(FU,T), U, C) \leftarrow select(\{completes(X)\}, F),\ par\_branch(X),\ findall(enables(X, S), (seq(X, S)), Enbls),\ update(F, \{completes(X)\}, Enbls, FU)$$

$$C3.\ tr(s(F,T), s(FU,T), U, C) \leftarrow select(\{completes(X)\}, F),\ not\_par\_branch(X), seq(X, S),\ update(F, \{completes(X)\}, \{enables(X, S)\}, FU)$$

$$C4.\ tr(s(F,T), s(FU,T), U, C) \leftarrow select(Enbls, F),\ par\_merge(X),\ findall(enables(P, X), (seq(P, X)), Enbls),\ update(F, Enbls, \{begins(X)\}, FU)$$

$$C5.\ tr(s(F,T), s(FU,T), U, C) \leftarrow select(\{enables(P, X)\}, F),\ not\_par\_merge(X),\ update(F, \{enables(P, X)\}, \{begins(X)\}, FU)$$

$$C6.\ tr(s(F,T), s(FU,T), U, C) \leftarrow select(\{enacting(X, R)\}, F),\ R=0,\ update(F, \{enacting(X, R)\}, \{completes(X)\}, FU)$$

$$C7.\ tr(s(F,T), s(FU,TU), U, C) \leftarrow no\_other\_premises(F),\ member(enacting(\_,\_), F),\ findall(Y, (Y = enacting(X, R),\ member(Y, F)), Enacts),\ mintime(Enacts, M),\ M > 0,\ decrease\_residual\_times(Enacts, M, EnactsU),\ findall(Z, (Z = enables(P, S), member(Z, F)), Enbls),\ set\_union(Enacts, Enbls, EnactsEnbls),\ update(F, EnactsEnbls, EnactsU, FU),\ TU = T + M$$

(1) Encoding of the semantics

reach: *reflexive, transitive closure of the transition relation* *tr*

$R1$: $\quad reach(S,S,U,C) \leftarrow$

$R2$: $\quad reach(S0,S2,U,C) \leftarrow tr(S0,S1,U,C),\ reach(S1,S2,U,C)$

(2) Encoding Reachability

- *Reachability Property.*

  $$RP: \quad \mathit{reachProp}(U,C) \leftarrow c(T,U,C), \quad \mathit{reach}(\mathit{init}, \mathit{fin}(T), U, C)$$

  where $c(T,U,C)$ is a constraint

- *Initial state.* $\mathit{init}: \quad < \{\mathit{begins}(\mathit{start})\}, 0 >$
- *Final state.* $\mathit{fin}(T): \quad < \{\mathit{completes}(\mathit{end})\}, T >$

(2) Encoding Controllability

Let *Sem* be the CHC encoding of semantics:

$C1$-$C7$ (for *tr*) and $R1$-$R2$ (for *reach*).

Let *LIA* be the theory of Linear Integer Arithmetics.

- *Weak Controllability*

$$Sem \cup \{RP\} \cup LIA \models \boxed{\forall U.\ adm(U) \rightarrow \exists C\ reachProp(U,C)}$$

where $adm(U)$ iff the durations in U belong to the given intervals

- *Strong Controllability*

$$Sem \cup \{RP\} \cup LIA \models \boxed{\exists C.\ \forall U.\ adm(U) \rightarrow reachProp(U,C)}$$

(3) Program Transformation

- Validity of Weak and Strong Controllabilities:
  - cannot be proved by CHC solvers over *LIA* (e.g., Z3), because of the complex terms (such as those denoting sets) and the *findall* predicate in *Sem*
  - cannot be proved by CLP systems, because of $\exists$−$\forall$ and $\forall$−$\exists$
  - solvers and CLP systems have termination problems due to recursive *reach.*

Solving Controllability Problems using **Constrained Horn Clauses (CHCs)**

- Constrained Horn Clauses: $H \leftarrow \boldsymbol{c},\ A_1, \ldots, A_n$
- Use Constrained Horn Clauses to:

(1) Encode the *semantics* of time-aware business processes

(2) Encode *reachability* and *controllability* properties.

(3) *Transformation of CHCs*

(4) Applying algorithms for controllability by *using CHC solvers,* (i.e., tools for *Satisfiability Modulo Theory* specialized to CHCs over integers).

(3) Program Transformation

- We transform *Sem* $\cup$ {*RP*} for removing complex terms/*findall* and derive equisatisfiable function-free, linear-recursive clauses:

  $$p(X) \leftarrow \boldsymbol{c},\ q(Y)$$

  where X, Y are tuples of variables and $\boldsymbol{c}$ is a constraint in *LIA*

RI: Removal of the Interpreter

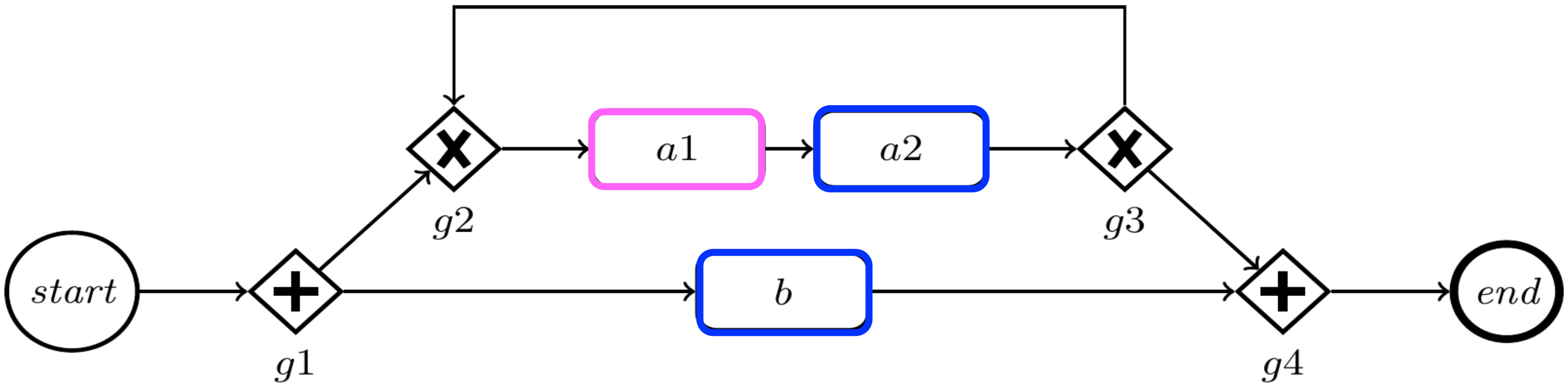
- Transformation rules: unfolding, definition, folding, clause removal
- Removal of the Interpreter (RI strategy): specialization of *Sem* with respect to the given business process and the given property *RP*

- $Sem \cup \{RP\} \xrightarrow{\text{RI}} I_{SP}$ s.t., for all $u,c \in \mathbf{N}$

$$\boxed{Sem \cup \{RP\} \cup LIA \vDash reachProp(u,c) \quad \text{iff} \quad I_{SP} \vDash reachProp(u,c)}$$

- I_{SP} is a set of function-free, linear-recursive clauses.

Removal of the Interpreter (example)



$event(start) \leftarrow$ $task(a1) \leftarrow$ $par\_branch(g1) \leftarrow$ ...

$seq(start, g1) \leftarrow$ $seq(g1, g2) \leftarrow$ $seq(g1, b) \leftarrow$ ...

$uncontrollable(a1) \leftarrow$ $controllable(a2) \leftarrow$ $controllable(b) \leftarrow$

$duration(g1, D) \leftarrow D = 0$ $duration(a1, D) \leftarrow 2 \leq D \leq 4$

$duration(a2, D) \leftarrow 1 \leq D \leq 2$ $duration(b, D) \leftarrow 5 \leq D \leq 6$ ...

RP: $reachProp(A1,(A2,B)) \leftarrow true, reach(init,fin(T), A1,(A2,B))$

Weak Controllability: $\forall A1.\ 2 \leq A1 \leq 4 \rightarrow \exists A2,B.\ reachProp(A1,(A2,B))$

Removal of the Interpreter (example)

- Fully automatic transformation using **VeriMAP** [DFPP-15].
  We get I_{SP} :

$$
\begin{array}{l}
reachProp(A1, (A2,B)) \leftarrow A{=}A1,\ B{=}B1,\ A1{\geq}2,\ A1{\leq}4,\ B{\geq}5,\ B{\leq}6, \\
\qquad\qquad\qquad\qquad\qquad\qquad new2(A, B1, F, G, A1, A2, B) \\
new2(A,B1,C,D,A1,A2,B) \leftarrow H{=}A{+}C,\ I{=}B1{-}A,\ J{=}0,\ A{\geq}1,\ I{\geq}0,\ A{+}I{\geq}1, \\
\qquad\qquad\qquad\qquad\qquad new2(J, I, H, D, A1, A2, B) \\
new2(A,B1,C,D,A1,A2,B) \leftarrow H{=}B1{+}C,\ I{=}A{-}B1,\ J{=}0,\ A{\geq}1,\ I{\geq}0,\ A{-}I{\geq}1, \\
\qquad\qquad\qquad\qquad\qquad new2(I,J,H,D,A1,A2,B) \\
new2(A,B1,C,D,A1,A2,B) \leftarrow H{=}A2,\ A{=}0,\ H{\geq}1,\ H{\leq}2,\ new5(H,B1,C,D,A1,A2,B) \\
new5(A,B1,C,C,A1,A2,B) \leftarrow A{=}0,\ B1{=}0 \\
new5(A,B1,C,D,A1,A2,B) \leftarrow H{=}A{+}C,\ I{=}B1{-}A,\ J{=}0,\ A{\geq}1,\ I{\geq}0,\ A{+}I{\geq}1,\ new5(J,I,H,D,A1,A2,B) \\
new5(A,B1,C,D,A1,A2,B) \leftarrow H{=}B1{+}C,\ I{=}A{-}B1,\ J{=}0,\ A{\geq}1,\ I{\geq}0,\ A{-}I{\geq}1,\ new5(I,J,H,D,A1,A2,B) \\
new5(A,B1,C,D,A1,A2,B) \leftarrow H{=}A1,\ A{=}0,\ H{\geq}2,\ H{\leq}4,\ new2(H,B1,C,D,A1,A2,B)
\end{array}
$$

- Function-free, linear recursive CHCs over the integers.

  But,... the CHC solver Z3 is still unable to prove Weak Controllability

  because of the recursive predicates *new2* and *new5*.

(4) Weak Controllability Algorithm

(1) Generate a disjunction $a(U,C)$ of constraints

(2) Check whether or not $LIA \models \forall U.\ adm(U) \rightarrow \exists C.\ a(U,C)$

- Assume a sound and complete LIA-constraint solver: SOLVE.
  For any set I_{SP} of clauses and query Q: $c, A_1,\ldots,A_n$
  where c is a LIA constraint,

  SOLVE(I_{SP},Q) returns

  - a satisfiable constraint a s.t. $I_{SP} \cup LIA \models \forall(a \rightarrow Q)$, if any,
  - *false*, otherwise

In particular, if SOLVE$(I_{SP},\ reachProp(U,C)) = a(U,C)$, then

$$I_{SP} \cup LIA \models \forall U,C.\ (a(U,C) \rightarrow reachProp(U,C))$$

(4) Weak Controllability Algorithm

I_{SP} : $\quad q(X) \leftarrow r(X)$

$\quad\quad r(X) \leftarrow X>0$

SOLVE(I_{SP}, q(X)) returns the constraint $X>0$

Indeed, $I_{SP} \cup LIA \models \forall X \; (X>0 \rightarrow q(X))$

(4) Weak Controllability Algorithm

$a(U,C) := \mathit{false};$

$\mathit{do}\ \{$

$\quad Q := (\mathrm{reachProp}(U,C) \land \forall C.\ \neg a(U,C));$

$\quad \mathit{if}\ (\mathrm{SOLVE}(I_{SP}, Q) = \mathit{false})\ \ \mathit{return}\ \ \mathit{false};$

$\quad a(U,C) := a(U,C) \lor \mathrm{SOLVE}(I_{SP}, Q);$

$\}\ \mathit{while}\ (\mathit{LIA} \not\models \forall U.\ \mathit{adm}(U) \rightarrow \exists C.\ a(U,C));$

$\mathit{return}\ a(U,C);$

(4) Weak Controllability Algorithm

```
a(U,C) := false;
do {
        Q := (reachProp(U,C) ∧ ∀C. ¬a(U,C));
        if (SOLVE(I_SP, Q) = false)  return  false;
        a(U,C) := a(U,C) ∨ SOLVE(I_SP,Q);
} while (LIA ⊭ ∀U. adm(U) → ∃C. a(U,C));
return a(U,C);
```

(4) Weak Controllability Algorithm

- By definition of SOLVE, the *do-while* constructs a sequence $<a_1(U,C),\ldots,a_n(U,C)>$ of *disjoint constraints* such that

$$\forall U,C.\ \ a_1(U,C) \rightarrow \text{reachProp}(U,C) \wedge \ldots \wedge a_n(U,C) \rightarrow \text{reachProp}(U,C)$$

that is, $\forall U,C.\ \ (a_1(U,C) \vee \ldots \vee a_n(U,C)) \rightarrow \text{reachProp}(U,C)$

(4) Weak Controllability Algorithm

- By definition of SOLVE, the *do-while* constructs a sequence $< a_1(U,C),\ldots,a_n(U,C) >$ of *disjoint constraints* such that

$$\forall U,C.\ \ a_1(U,C) \rightarrow \mathrm{reachProp}(U,C) \ \wedge \ \ldots \ \wedge \ a_n(U,C) \rightarrow \mathrm{reachProp}(U,C)$$

that is, $\forall U,C.\ (a_1(U,C) \vee \ldots \vee a_n(U,C)) \rightarrow \mathrm{reachProp}(U,C)$ (1)

- Moreover, at termination of the *do-while*:

$$\forall U.\ adm(U) \rightarrow \exists C.\ (a_1(U,C) \vee \ldots \vee a_n(U,C)) \qquad (2)$$

- From (1) and (2), by transitivity we get:

$$\forall U.\ adm(U) \rightarrow \exists C.\ reachProp(U,C) \qquad \text{(weak controllability)}$$

as desired.

(4) Weak Controllability Algorithm

$a(U,C) := \mathit{false};$
do {
  $Q := (\mathrm{reachProp}(U,C) \wedge \forall C.\ \neg a(U,C));$
  $\text{if } (\mathrm{SOLVE}(I_{SP}, Q) = \mathit{false})\ \ \mathit{return}\ \ \mathit{false};$
  $a(U,C) := a(U,C) \vee \mathrm{SOLVE}(I_{SP}, Q);$
$\}\ \mathit{while}\ (LIA \not\models \forall U.\ adm(U) \rightarrow \exists C.\ a(U,C));$
$\mathit{return}\ a(U,C);$

1) $\mathrm{SOLVE}(I_{SP}, \mathrm{reachProp}(A1, (A2, B)) \wedge \neg\, \mathit{false})$
returns $a1(A1, A2, B)$, which is $B-2 \leq A1 \leq 4 \wedge A2 = B-A1 \wedge 5 \leq B \leq 6.$
$LIA \not\models \forall A1.\ 2 \leq A1 \leq 4 \rightarrow \exists A2, B.\ a1(A1, A2, B)$

(4) Weak Controllability Algorithm

$a(U,C) := \mathit{false};$
do {
  $Q := (\mathrm{reachProp}(U,C) \land \forall C.\ \neg a(U,C));$
  if $(\mathrm{SOLVE}(I_{SP}, Q) = \mathit{false})$ *return false*;
  $a(U,C) := a(U,C) \lor \mathrm{SOLVE}(I_{SP}, Q);$
} *while* $(LIA \not\models \forall U.\ adm(U) \rightarrow \exists C.\ a(U,C));$
return $a(U,C);$

1) $\mathrm{SOLVE}(I_{SP}, \mathrm{reachProp}(A1, (A2, B)) \land \neg \mathit{false})$
   returns $a1(A1, A2, B)$, which is $B-2 \le A1 \le 4 \land A2=B-A1 \land 5 \le B \le 6$.
   $LIA \not\models \forall A1.\ 2 \le A1 \le 4 \rightarrow \exists A2,B.\ a1(A1, A2, B)$

2) $\mathrm{SOLVE}(I_{SP}, \mathrm{reachProp}(A1, (A2, B)) \land \forall A2,B.\ \neg a1(A1, A2, B))$
   returns $a2(A1, A2, B)$, which is $A1=2 \land A2=1 \land B=6$.
   $LIA \models \forall A1.\ 2 \le A1 \le 4 \rightarrow \exists A2,B.\ (a1(A1, A2, B) \lor a2(A1, A2, B))$

(4) Strong Controllability Algorithm

$a(U,C) := \mathit{false}$

do {

$\quad Q := (\mathrm{reachProp}(U,C) \wedge \neg a(U,C));$

$\quad \mathit{if}\ (\mathrm{SOLVE}(I_{SP}, Q) = \mathit{false})\ \ \mathit{return}\ \ \mathit{false};$

$\quad a(U,C) := a(U,C) \vee \mathrm{SOLVE}(I_{SP}, Q);$

} $\mathit{while}\ (\mathit{LIA} \not\models \exists C. \forall U.\ \mathit{adm}(U) \rightarrow a(U,C));$

$\mathit{return}\ a(U,C);$

Implementation

- Different tools have been used:
  - **VeriMAP** transformation system for RI (Removal of the Interpreter)
  - **SICStus** Prolog: Computation of answer constraints
  - **Z3**: SMT solver for checking quantified *LIA* formulas

Experimental evaluation

Experimentation on various examples:

- Purchase order [DFMPP 2016]
- Request Day-Off Approval [Huai et al. 2010]
- STEMI: Emergency Department Admission [Combi et al. 2009]
- STEMI: Emergency Department + Coronary Care Unit Admission [Combi et al. 2012]

Conclusions

- Controllability was introduced in various contexts
  [Vidal-Fargier 1999, Combi-Posenato 2009, Cimatti et al. 2015, Zavatteri et al. 2017]
- We presented a flexible framework for reasoning about controllability
  - parametric with respect to the semantics and the property
  - use of satisfiability-preserving CHC transformations
  - use of state-of-the-art CHC solvers and CLP systems
- Possible future developments:
  - Larger fragment of BPMN: timers, interrupting events, ...
  - Data [Montali et al. 2013, Deutsch 2014, ...]
  - Ontologies for tasks, ...

Many thanks for the invitation

... and many thanks also to Professor Rasiowa.
She has a special place in my heart.

The end.