Verifying Controllability of Time-Aware Business Processes

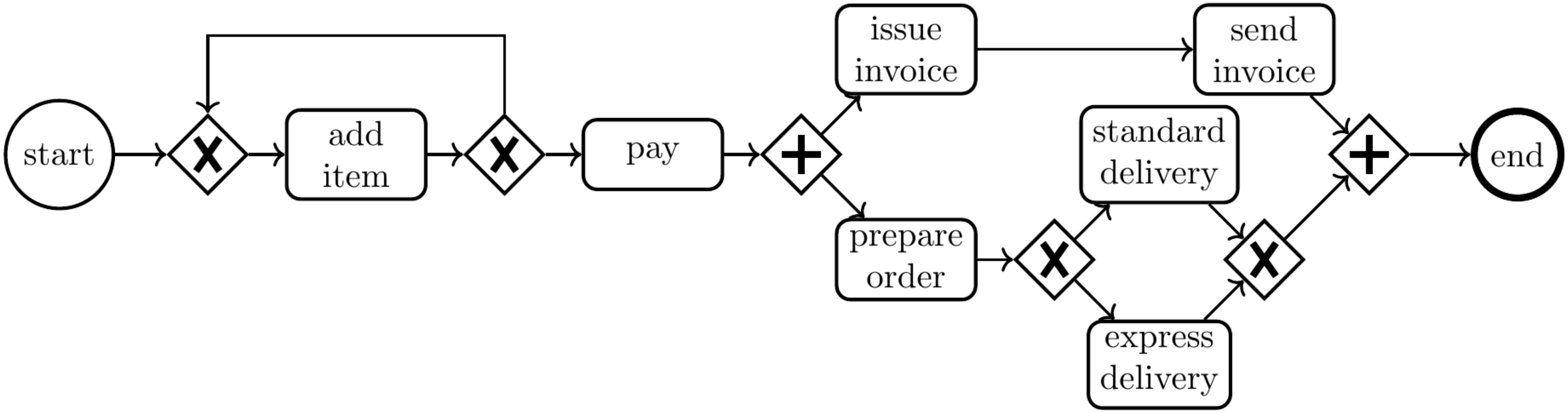
E. De Angelis [1], F. Fioravanti [1], M.C. Meo [1]
A. Pettorossi [2], *M. Proietti* [3]

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

RuleML+RR 2017 **London (UK) July 12, 2017**

Business Processes

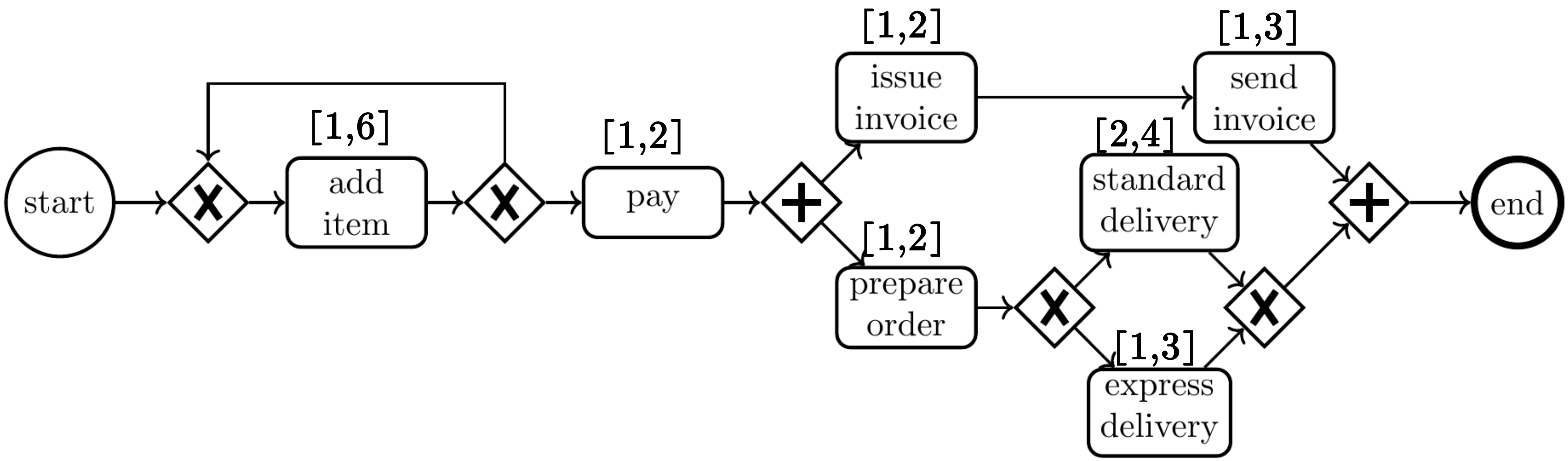
- A process that *coordinates the activities of an organization* towards a business goal
- *Purchase Order*: A customer adds one or more items to the shopping cart and pays. Then, the vendor sends the invoice and delivers the order



- No quantitative time information (e.g., durations of tasks)

Time-aware Business Processes

- Specify *intervals* of task duration: $\mathbf{D} \in [\mathbf{dmin}, \mathbf{dmax}] \subset \mathbb{N}$

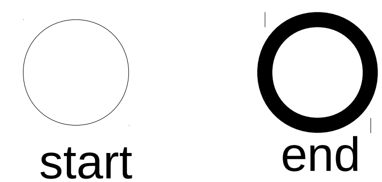


- *Reachability* property: The time to reach ‘end’ from ‘start’ is less than K
- *Controllability* property: It is possible to determine the durations of some (controllable) tasks so that a given reachability property holds

Business Process Modeling and Notation (BPMN)

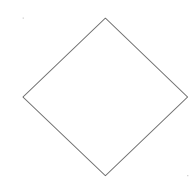
- *Graphical language* for modeling organizational processes: activities, events, and their composition (OMG standard)
- *Tasks*: atomic activities

- *Events*: something that ‘happens’



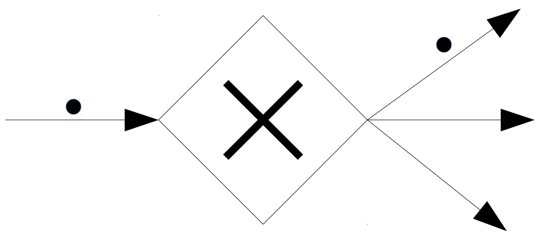
- *Sequence flow*: order of execution

- *Gateways*: branching/merging flows

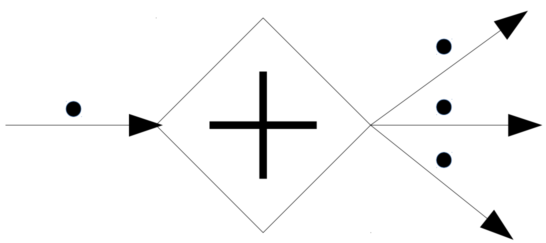


Branch gateways

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

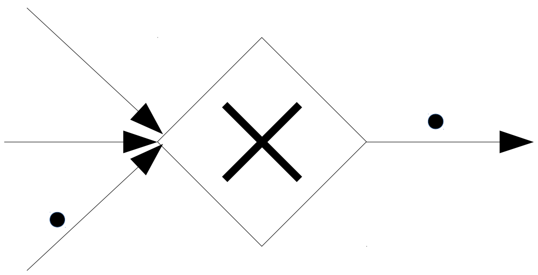


- **parallel** branch gateway (AND)
  - upon activation
    of the incoming flow
    all outgoing flows
    are activated

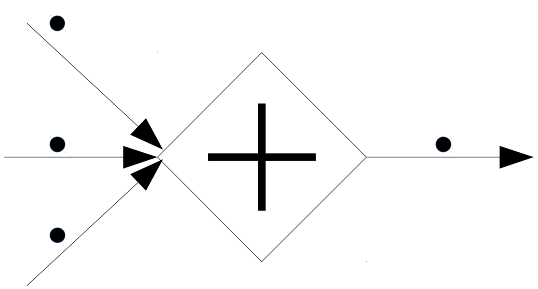


Merge gateways

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



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



Semantics of time-aware BPMN

- Transition relation → between states <*F*,*t*>
- *t* time point: non-negative integer
- *F* set of *fluents*: properties that hold at time point *t*
  - *begins*(*x*): *x* begins its execution (enactment)
  - *enacting(x,r):* *x* is enacting, *r* residual time to completion
  - *completes*(*x*): *x* completes its execution
  - *enables*(*x,y*): *x* enables its successor *y*

  x,y denote flow objects (tasks, events, or gateways)
- *seq*(*x,y*): there is a sequence flow from *x* to *y*
- *duration*(*x*,*d*): the duration of *x* is *d*

Semantics of time-aware BPMN

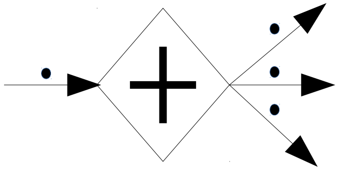
- Instantaneous transitions

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

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

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

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



Semantics of time-aware BPMN

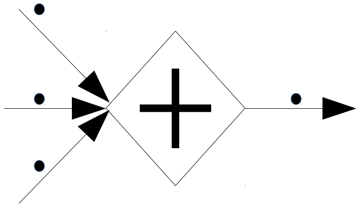
- Instantaneous transitions

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

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

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

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



Semantics of time-aware BPMN

- Time elapsing transition

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

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) $Enbls = \{enables(p,s) \mid enables(p,s) \in F\}$.

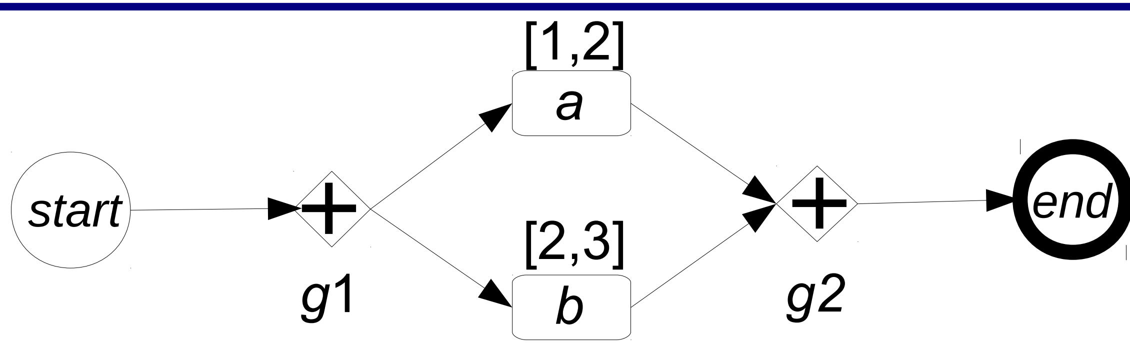
- Time elapses when no istantaneous transition can occur. All enacting tasks proceed in parallel for a time equal to the minimum of all residual times.

Reachability

- State $<F,t>$ is reachable iff, *for some durations in the given intervals*,

$$<\{\mathit{begins}(\mathit{start})\},0> \;\rightarrow^{*}\; <F,t>$$

An example of enactment



<{begins(start)},0> $\rightarrow^*$ <{begins(g1)}, 0>

(S_1) $\rightarrow$ <{enacting(g1,0)}, 0> % duration(g1,0)

(S_6) $\rightarrow$ <{completes(g1)}, 0>

(S_2) $\rightarrow$ <{enables(g1,a),enables(g1,b)}, 0>

(S_5) $\rightarrow$ <{begins(a),enables(g1,b)}, 0>

(S_1) $\rightarrow$ <{enacting(a,2),enables(g1,b)}, 0> % 2 in [1,2]

(S_5S_1) $\rightarrow^2$ <{enacting(a,2),enacting(b,2)}, 0> % 2 in [2,3]

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

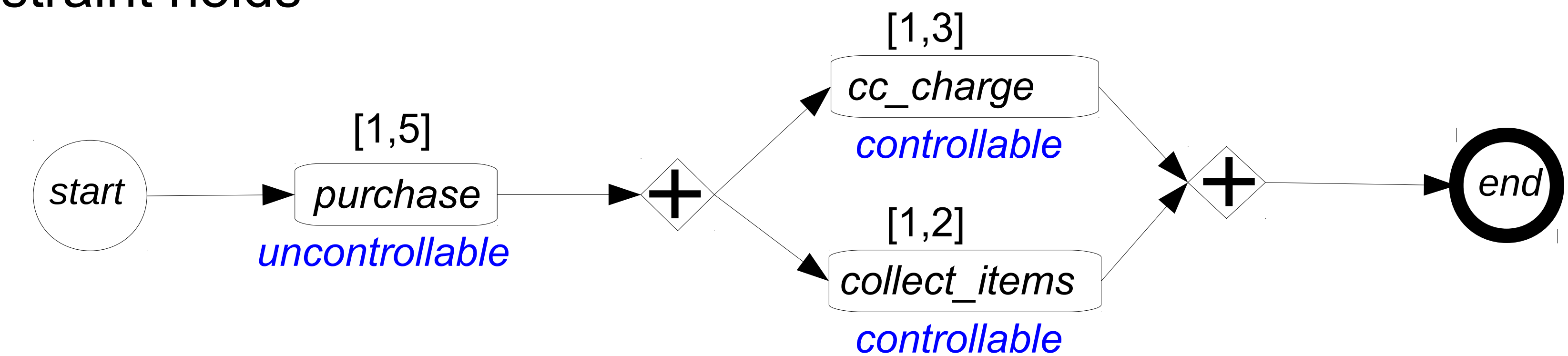
(S_6S_6) $\rightarrow^2$ <{completes(a),completes(b)}, 2>

(S_3S_3) $\rightarrow^2$ <{enables(a,g2),enables(b,g2)}, 2>

(S_4) $\rightarrow$ <{begins(g2)}, 2>

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

Weak Controllability

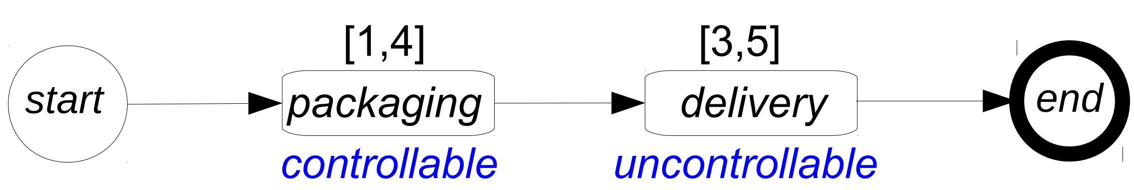
- Assume:
  - Some tasks are *controllable* (e.g., internal to the organization)
  - Some tasks are *uncontrollable* (e.g., external to the organization)
- WC: *For all durations of the uncontrollable tasks* (within the given time intervals), we can *determine durations of the controllable tasks* (within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

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

a solution: *if* $D_{pu}=1$ *then* $D_{cc}=D_{ci}=2$ *else* $D_{cc}=D_{ci}=1$

Strong Controllability

- WC may not be useful when some uncontrollable tasks occur *after* controllable ones
- SC: 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 holds
- The exact duration of the delivery is not known when packaging



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

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

Solving Controllability Problems with Constrained Horn Clauses

- Constrained Horn Clauses (aka CLP): $H \leftarrow c, A_1, \ldots, A_n$
- Use Constrained Horn Clauses to:

1) Encode the *semantics* of time-aware BPs;
2) Encode *reachability* and *controllability* problems;
3) Solve controllability problems by applying *CHC solvers* (i.e., tools for *Satisfiability Modulo Theory* specialized to CHCs over integers).

1) CHC encoding of the semantic rules

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

$$\Downarrow$$

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

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

CHC interpreter for time-aware BPMN

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

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

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

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

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

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

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

2.1) Encoding reachability

- *Reachability*:

  $R1$: $\quad reach(S,S,U,C) \leftarrow$
  $R2$: $\quad reach(S0,S2,U,C) \leftarrow tr(S0,S1,U,C),\ reach(S1,S2,U,C)$

- *Reachability Property*:

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

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

- *Initial state* *init*: $\langle\{begins(start)\},0\rangle$,
- *Final state* *fin*(T): $\langle\{completes(end)\},T\rangle$
- Similarly for non final states

2.2) Encoding controllability

- CHCs *Sem* encoding of semantics of a BP: clauses C1-C7,R1,R2
- Theory of Linear Integer Arithmetics (*LIA*)
- *Weak Controllability*:

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

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

- *Strong Controllability*:

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

3) Applying CHC solvers

- Validity of WC and SC properties:
  - cannot be proved by CHC solvers over *LIA* (e.g., Z3), because of complex terms (e.g., {.}), findall predicate in the interpreter
  - cannot be proved by CLP systems, because of ∃∀ and ∀∃
  - both may have termination problems with recursive reach

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

  $$p(X) \leftarrow c,\ q(Y)$$

  where X, Y are tuples of variables and c is a constraint in *LIA*

Removal of the interpreter

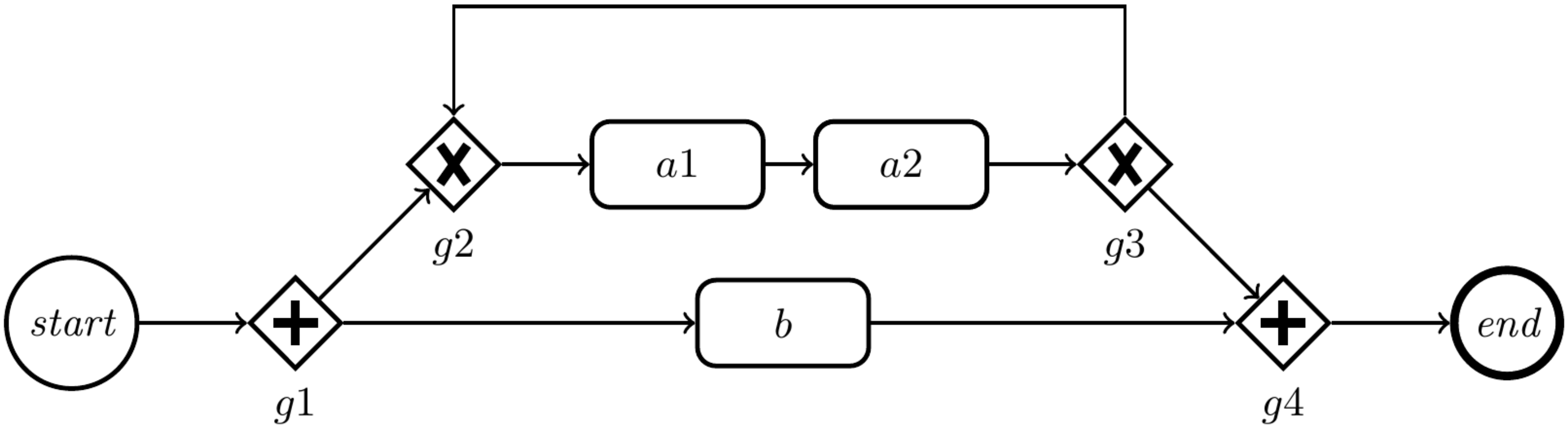
- Rule-based transformation strategy
  - Transformation rules: unfolding, definition, folding, clause removal
  - Strategy: specialization of the interpreter *Sem* with respect to the specific business process and the specific property *RP*

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

  $Sem \cup \{RP\} \cup LIA \models reachProp(u,c)$ iff $I_{SP} \models reachProp(u,c)$

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

Removal of the Interpreter: Example



task(*a*1) ← *event*(*start*) ← *par_branch*(*g*1) ← ...

seq(*start*, *g*1) ← *seq*(*g*1, *b*) ← ...

uncontrollable(*a*1) ← *controllable*(*a*2) ← *controllable*(*b*) ←

duration(*a*1, *D*) ← $2 \leq D \leq 4$ *duration*(*a*2, *D*) ← $1 \leq D \leq 2$

duration(*b*, *D*) ← $5 \leq D \leq 6$ *duration*(*g*1, *D*) ← $D = 0$ …

RP: *reachProp*(*A*1,*A*2,*B*) ← *reach*(*init*,*fin*(*T*),*A*1,*A*2,*B*)

WC: $\forall A1.\ 2 \leq A1 \leq 4 \rightarrow \exists A2,B.$ *reachProp*(*A*1,*A*2,*B*)

... Example

- Fully automatic transformation using ***VeriMAP*** [DFPP-15]

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

- Function-free, linear recursive CHCs over the integers
- The CHC solver Z3 is still unable to prove WC because of ∀ over the recursively defined predicates *new*2 and *new*5

Controllability Algorithms

- Reduce verification to solving quantified *non-recursive LIA* formulas
- We assume we have a *solver* SOLVE. For any set P of clauses and query Q: $c, A_1, \ldots, A_n$,

  SOLVE(P,Q) returns

  - a satisfiable *answer constraint a* s.t. $P \cup LIA \models \forall(a \rightarrow Q)$, if any
  - *false*, otherwise

WC 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);
```

WC 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);

1) SOLVE(P, reachProp(A1, A2, B) ∧ ¬false) =
a1(A1, A2, B): B−2≤A1≤4 ∧ A2=B−A1 ∧ 5≤B≤6

LIA ⊭ ∀A1. 2≤A1≤4 → ∃A2,B. a1(A1, A2, B)

WC 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);

1) SOLVE(P, reachProp(A1, A2, B) ∧ ¬false) =
a1(A1, A2, B): B−2≤A1≤4 ∧ A2=B−A1 ∧ 5≤B≤6
LIA ⊭ ∀A1. 2≤A1≤4 → ∃A2,B. a1(A1, A2, B)

2) SOLVE(P, reachProp(A1, A2, B) ∧ ∀A2,B. ¬a1(A1, A2, B)) =
a2(A1, A2, B): A1=2 ∧ A2=1 ∧ B=6
LIA ⊨ ∀A1. 2≤A1≤4 → ∃A2,B. (a1(A1, A2, B) ∨ a2(A1, A2, B))

SC Algorithm

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

Implementation

- Different tools have been used to implement the technique:
  - VeriMAP transformation system: Removal of the Interpreter
  - SICStus Prolog: Computation of answer constraints
  - Z3 SMT solver: Checking quantified LIA formulas

- Integration is underway

Experimental evaluation

- Experimentation in progress 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 introduced in various contexts [VidalFargier-99,CimattiEtAl-15,CombiPosenato-09]
- This talk: Flexible framework for reasoning about the controllability of time-aware BPs
  - Parametric w.r.t. the semantics and property
  - Satisfiability-preserving CHC transformations
  - State-of-the-art CHC solvers and CLP systems

- Future developments
  - larger fragment of BPMN (timer events)
  - data (Montali, Deutsch, ...)
  - Ontologies (Semantic BP models)

The end

Thank you!