Verifying Controllability of Time-Aware Business Processes

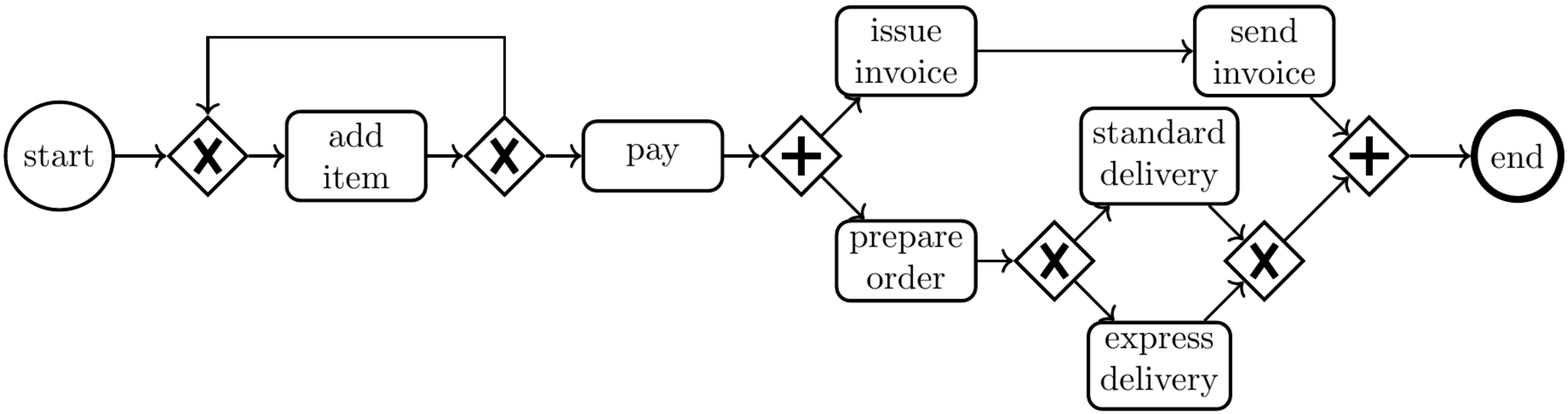
E. De Angelis [1], F. Fioravanti [1], M.C. Meo [1]
A. Pettorossi [2], *M. Proietti* [3]

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

CILC 2017 **Napoli, 27 Settembre 2017**

Business Processes

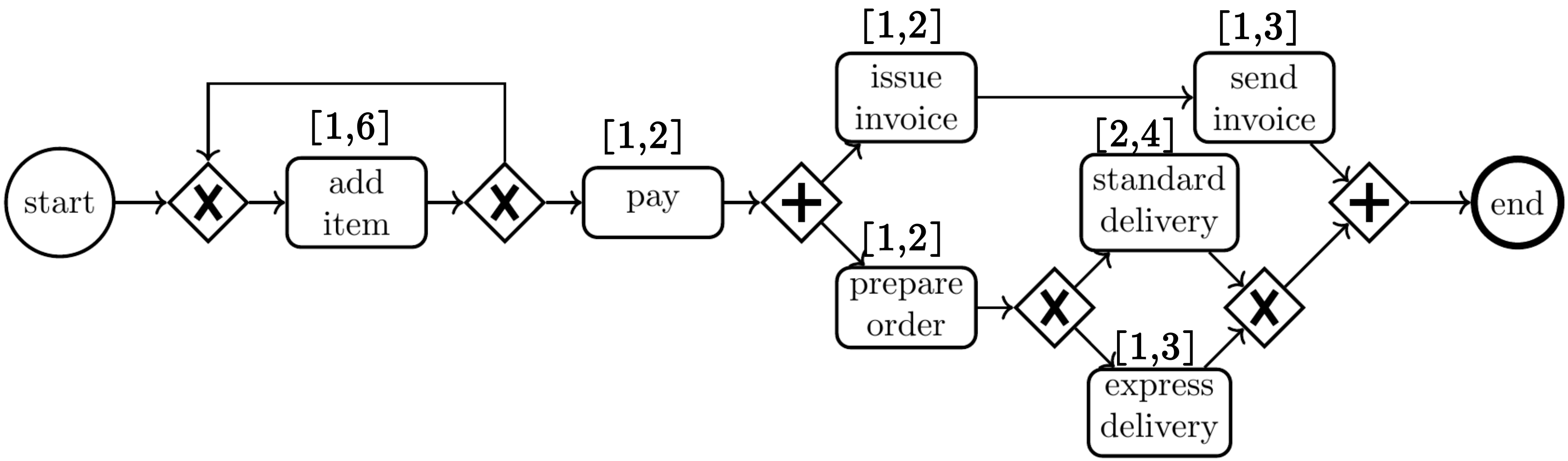
- A BP is a set of activities and tasks that need to be accomplished to deliver a service or product
- *Purchase Order*: A customer adds one or more items to the shopping cart and pays. Then, the vendor sends the invoice and delivers the order



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

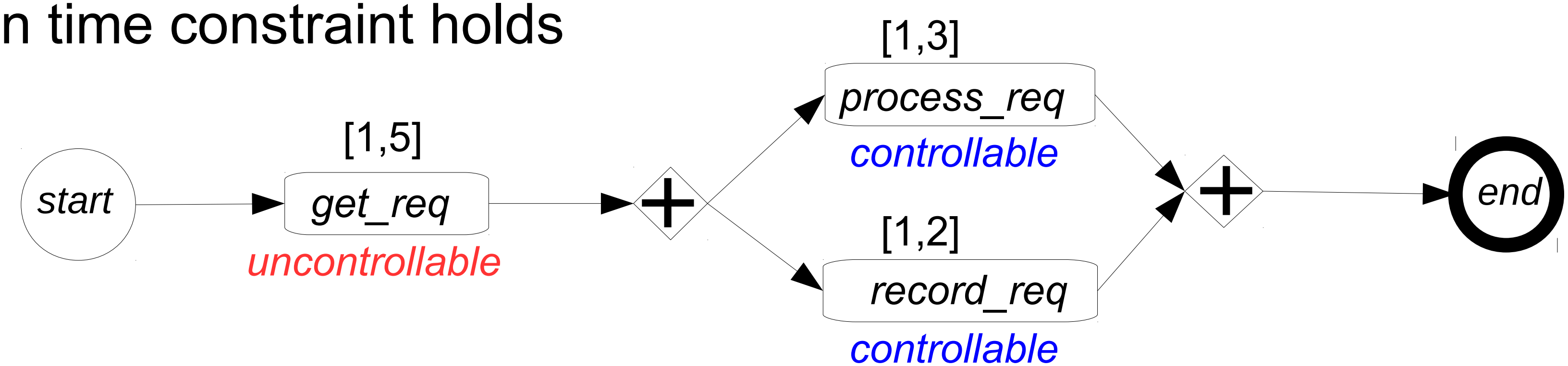
Time-Aware Business Processes

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



- *Reachability* property: The time to reach ‘end’ from ‘start’ satisfies a given constraint
- *Controllability* property: It is possible to determine the durations of some tasks so that a given reachability property holds

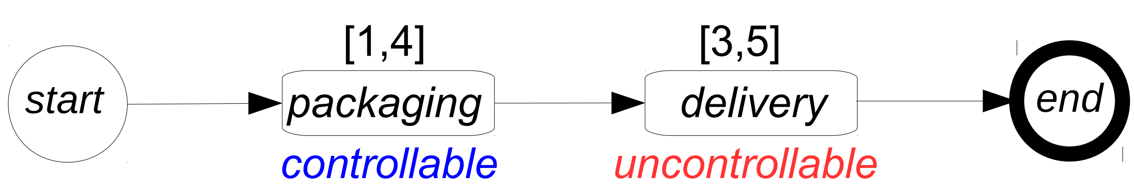
Weak Controllability

- 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. the process can be completed and a given time constraint holds
- 

WC: ∀ durations of *get_req* in $[1,5]$, ∃ durations of *process_req* in $[1,3]$ and *record_req* in $[1,2]$, such that $3 \leq t_{total} \leq 7$

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), the process can be completed and a given time constraint holds
- The exact duration of the delivery is not known when packaging

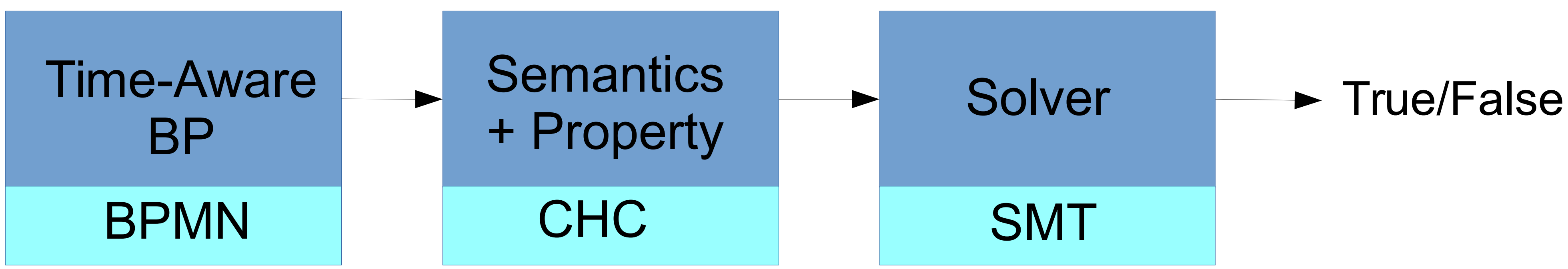


∃ durations of *packaging* in $[1,4]$ such that, ∀ durations of *delivery* in $[3,5]$, the constraint $4 \leq t_{total} \leq 7$ holds

Verifying Time-Aware BPs using Constrained Horn Clauses

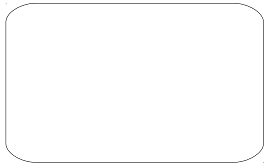
Use *Constrained Horn Clauses* (aka CLP) to:

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).

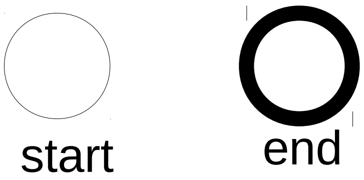


Business Process Modeling and Notation (BPMN)

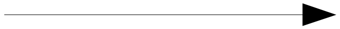
- *Graphical language* for modeling business processes: activities, events, and their order of execution (OMG standard)
- *Tasks*: atomic activities



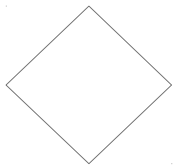
- *Events*: something that 'happens'



- *Sequence flow*: order of execution

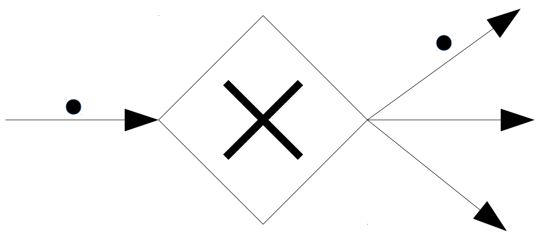


- *Gateways*: branching/merging of flows

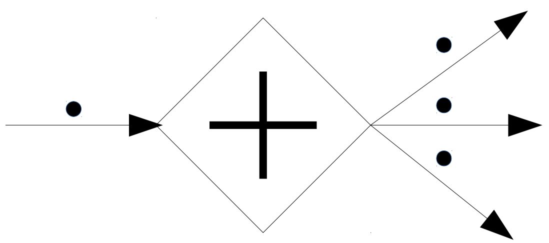


Branch Gateways

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

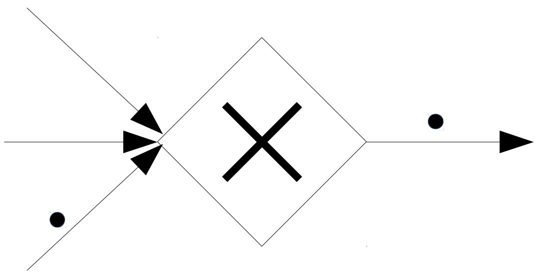


- **parallel** branch gateway (AND)
  - upon activation
    of the incoming flow
    <u>all</u> outgoing flows
    are instantaneously activated

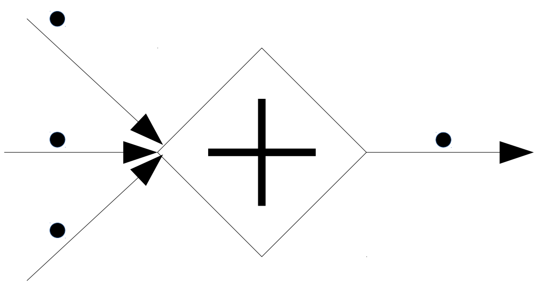


Merge gateways

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



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



1) Semantics of Time-Aware BPMN

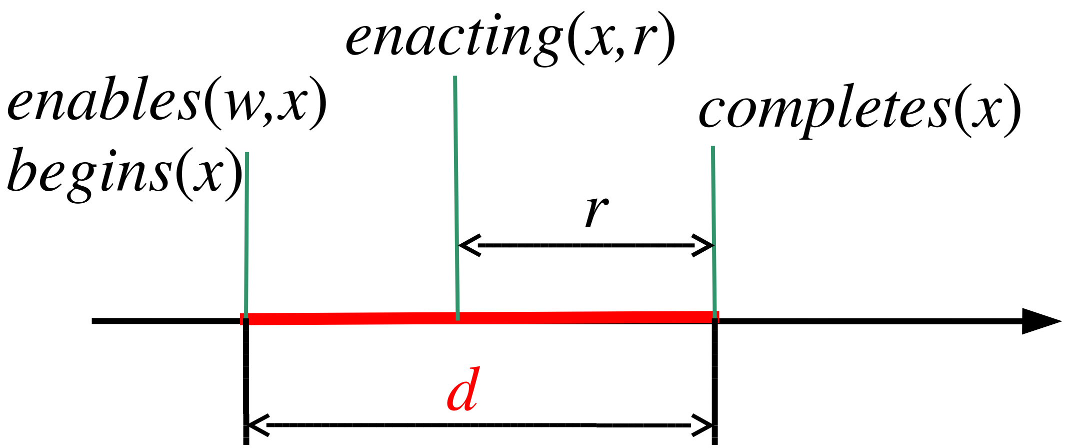
- *Transition relation* → between states $<F,t>$
- t time point: non-negative integer
- F set of *fluents*: properties that hold at time point t
  - $begins(x)$: x begins its execution (enactment)
  - $enacting(x,r)$: x is enacting, r residual time to completion
  - $completes(x)$: x completes its execution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

$task(w) \leftarrow \quad task(x) \leftarrow$

$duration(x, d) \leftarrow dmin \leq d \leq dmax$



- *Instantaneous* transitions: $<F,t> \rightarrow <F',t>$,
  e.g., $<\{begins(x), \ldots\}, t> \rightarrow <\{enacting(x,d), \ldots\}, t>$
- *Time-elapsing* transitions: $<F,t> \rightarrow <F',t'>$,
  e.g., $<\{enacting(x,r), \ldots\}, t> \rightarrow <\{enacting(x,0), \ldots\}, t+r>$

CHC Encoding of Semantics

Transition relation $S1 \rightarrow S2$ *encoded by a predicate*

$$tr(S1,S2,U,C)$$

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

CHC Semantics of Time-Aware BPMN

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

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

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

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

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

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

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

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 (for final state)*:

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

  where $c(T,U,C)$ is a constraint on time and durations

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

2.2) Encoding Controllability

- *Sem:* clauses C1-C7,R1,R2 encoding of semantics of a BP
- *LIA:* Theory of Linear Integer Arithmetic
- *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

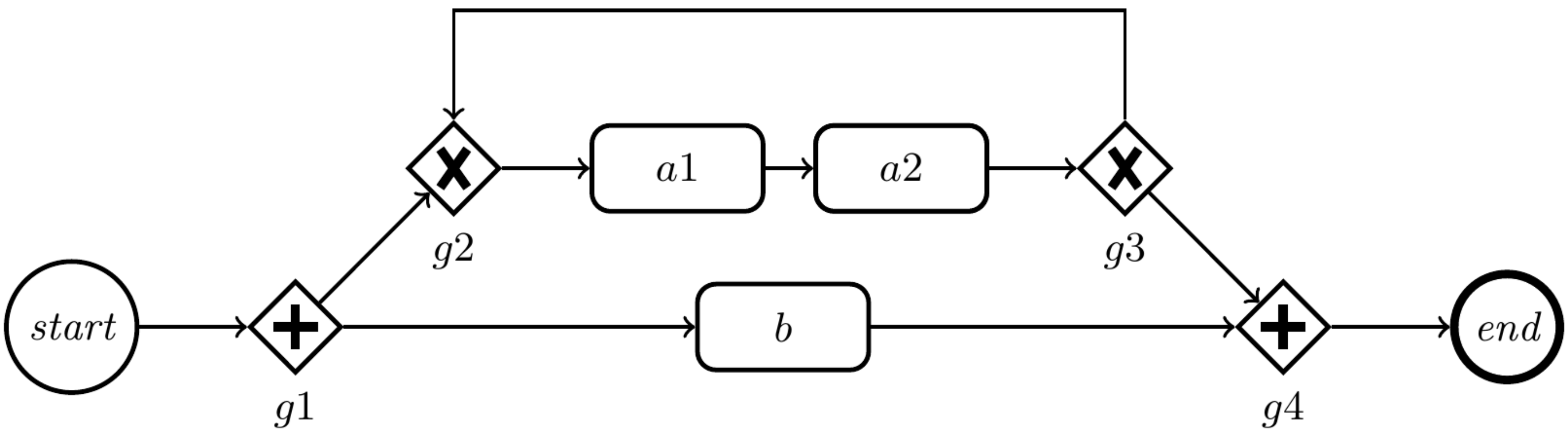
- *Transform* $Sem \cup \{RP\}$ for removing complex terms/findall and derive equisatisfiable *function-free*, *linear-recursive* clauses

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

  where X, Y are tuples of variables and c is a constraint in *LIA.*
  The transformation uses unfold/fold rules and *specializes* Sem to the specific business process and property RP

- Apply algorithms that reduce verification to solving sequences of ($\exists\forall$ and $\forall\exists$) quantified *non-recursive* *LIA* formulas

Transformation: Example



$task(a1) \leftarrow$ $event(start) \leftarrow$ $par\_branch(g1) \leftarrow$ ...
$seq(start, g1) \leftarrow$ $seq(g1, b) \leftarrow$ ...
$uncontrollable(a1) \leftarrow$ $controllable(a2) \leftarrow$ $controllable(b) \leftarrow$
$duration(a1, D) \leftarrow 2 \leq D \leq 4$ $duration(a2, D) \leftarrow 1 \leq D \leq 2$
$duration(b, D) \leftarrow 5 \leq D \leq 6$ $duration(g1, D) \leftarrow D = 0$ ...

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

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

... Example

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

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

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

CHC Solver

The controllbility algorithms use a *solver* SOLVE that, 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

Weak Controllability Algorithm

1) Generate a *disjunction* $a(U,C)$ of answer constraints
2) Check if $LIA \models \forall U.\ adm(U) \rightarrow \exists C.\ a(U,C)$ holds

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

Strong Controllability Algorithm

1) Generate a *disjunction* $a(U,C)$ of answer constraints
2) Check if $LIA \models \exists C\, \forall U.\ adm(U) \rightarrow a(U,C)$ holds

```
a(U,C) := false
do {
    Q := (reachProp(U,C) ∧ ¬a(U,C));
    if (SOLVE(P,Q) = false) return false;
    a(U,C) := a(U,C) ∨ SOLVE(P,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: Specialization of the Interpreter
  - SICStus Prolog: Computation of answer constraints
  - Z3 SMT solver: Checking quantified LIA formulas

- Integration is underway

Conclusions

- Controllability introduced in various contexts [VidalFargier-99,CimattiEtAl-15,CombiPosenato-09,CombiEtAl-17]
- 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 (Deutsch, Montali, ...)
  - domain-specific semantics (Ontologies)