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 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:  \( d \in [d_{\text{min}}, d_{\text{max}}] \subset \mathbb{N} \)

**Two problems:**

- **Time-Reachability:** checking whether or not to go from \( s \) to \( e \) takes less than \( k \) units of time.
**Information on the duration**: Intervals: \( d \in [d_{\text{min}}, d_{\text{max}}] \subset \mathbb{N} \)

**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.
**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
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 \( t \))
  - \( \text{begins}(x) \): \( x \) begins its execution (enactment)
  - \( \text{enacting}(x, r) \): \( x \) is executing with \( r \) residual time to completion
  - \( \text{completes}(x) \): \( x \) completes its execution
  - \( \text{enables}(x, y) \): \( x \) enables its successor \( y \)
  - \( x, y \) denote either tasks, or events, or gateways

- \( \text{seq}(x, y) \): there is an arrow from \( x \) to \( y \)

- \( t \) : time point (i.e., a non-negative integer)
  - \( \text{duration}(x, d) \): the duration of \( x \) is \( d \)
Semantics of time-aware BPMN

\[
\text{task}(x) \leftarrow
\]

\[
\text{duration}(x, d) \leftarrow 3 \leq d \leq 4
\]

\[
\text{enacting}(x, r) \text{ with } 0 \leq r \leq d
\]

- durations of events and gateways are assumed to be 0
Semantics of time-aware BPMN

Instantaneous transition:

\[ \text{begins}(x) \longrightarrow \text{enacting}(x,d) \]

\[ \langle F, t \rangle \longrightarrow \langle (F \setminus \{\text{begins}(x)\}) \cup \{\text{enacting}(x,d)\}, t \rangle \]
Semantics of time-aware BPMN

Instantaneous transition:

\[ \text{enacting}(x,0) \rightarrow \text{completes}(x) \]

\[ < F,t > \rightarrow < F',t > \]

\[ (S_6) \]

\[ \text{enacting}(x,0) \in F \]

\[ \langle F, t \rangle \rightarrow \langle (F \setminus \{\text{enacting}(x,0)\}) \cup \{\text{completes}(x)\}, t \rangle \]
Semantics of time-aware BPMN

Instantaneous transitions:

\[ < F, t > \rightarrow < F', t > \]

\[
(S_2) \quad \frac{\text{completes}(x) \in F \quad \text{par\_branch}(x)}{\langle F, t \rangle \rightarrow \langle (F \setminus \{\text{completes}(x)\}) \cup \{\text{enables}(x, s) \mid \text{seq}(x, s)\}, t \rangle}
\]

\[
(S_3) \quad \frac{\text{completes}(x) \in F \quad \text{not\_par\_branch}(x) \quad \text{seq}(x, s)}{\langle F, t \rangle \rightarrow \langle (F \setminus \{\text{completes}(x)\}) \cup \{\text{enables}(x, s)\}, t \rangle}
\]

\((S_2)\) If the parallel branch \(x\) completes, then \(x\) enables instantaneously all successors \(s\) of \(x\)
Semantics of time-aware BPMN

Instantaneous transitions:

\[ < F, t > \rightarrow < F', t > \]

\[(S_2) \quad \frac{completes(x) \in F \quad \text{par\_branch}(x)}{\langle F, t \rangle \rightarrow \langle (F \setminus \{completes(x)\}) \cup \{\text{enables}(x, s) \mid \text{seq}(x, s)\}, t \rangle} \]

\[(S_3) \quad \frac{completes(x) \in F \quad \text{not\_par\_branch}(x) \quad \text{seq}(x, s)}{\langle F, t \rangle \rightarrow \langle (F \setminus \{completes(x)\}) \cup \{\text{enables}(x, s)\}, t \rangle} \]

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

CS&P 2017 - Warsaw (Poland)
Semantics of time-aware BPMN

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

\[(S_4)\] If all predecessors $p$ of the parallel merge $x$ enable $x$, then the execution of $x$ begins instantaneously.

\[\forall p \text{ seq}(p, x) \rightarrow \text{enables}(p, x) \in F \quad \text{par_merge}(x)\]

\[\langle F, t \rangle \rightarrow \langle (F \setminus \{\text{enables}(p, x) \mid \text{enables}(p, x) \in F\}) \cup \{\text{begins}(x)\}, t \rangle\]

\[\text{enables}(p, x) \in F \quad \text{not_par_merge}(x)\]

\[\langle F, t \rangle \rightarrow \langle (F \setminus \{\text{enables}(p, x)\}) \cup \{\text{begins}(x)\}, t \rangle\]
Semantics of time-aware BPMN

Instantaneous transitions:

\[
(S_4) \quad \forall p \ seq(p, x) \rightarrow \text{enables}(p, x) \in F \quad \text{par}_\text{-merge}(x) \\
\langle F, t \rangle \rightarrow \langle (F \setminus \{\text{enables}(p, x) \mid \text{enables}(p, x) \in F\}) \cup \{\text{begins}(x)\}, \ t \rangle
\]

\[
(S_5) \quad \text{enables}(p, x) \in F \quad \text{not}_\text{-par}_\text{-merge}(x) \\
\langle F, t \rangle \rightarrow \langle (F \setminus \{\text{enables}(p, x)\}) \cup \{\text{begins}(x)\}, \ t \rangle
\]

\(S_4\) If all predecessors \(p\) of the parallel merge \(x\) enable \(x\), then the execution of \(x\) begins instantaneously.
Semantics of time-aware BPMN

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

\[
(S_7) \quad \frac{\text{no\_other\_premises}(F') \quad \exists x \exists r \text{ enacting}(x, r) \in F \quad m > 0}{\langle F, t \rangle \rightarrow \langle F \text{\ominus} m \setminus \text{Enbls}, t + m \rangle}
\]

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

Time elapses when no instantaneous 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

\[ F = \{ \text{enacting}(a, 5), \text{enacting}(b, 3) \} \]

at time: \( T \)

\[ F = \{ \text{enacting}(a, 2), \text{enacting}(b, 0) \} \]

at time: \( T + 3 \)

\[ < F, t > \rightarrow < F', t' > \]
Reachability

State $<F,t>$ is reachable iff

for some durations in the given intervals,\n
$$<\{\text{begins}(\text{start})\},0> \rightarrow^* <F,t>$$
Semantics in action

\[
\begin{align*}
\langle \text{begins}(\text{start}), 0 \rangle & \rightarrow^* \langle \text{begins}(g1), 0 \rangle \\
(S_1) & \rightarrow \langle \text{enacting}(g1, 0), 0 \rangle \\
(S_6) & \rightarrow \langle \text{completes}(g1), 0 \rangle \\
(S_2) & \rightarrow \langle \text{enables}(g1, a), \text{enables}(g1, b), 0 \rangle \\
(S_5) & \rightarrow \langle \text{begins}(a), \text{enables}(g1, b), 0 \rangle \\
(S_1) & \rightarrow \langle \text{enacting}(a, 2), \text{enables}(g1, b), 0 \rangle \\
(S_5 S_1)^2 & \rightarrow \langle \text{enacting}(a, 2), \text{enacting}(b, 2), 0 \rangle \\
(S_7) & \rightarrow \langle \text{enacting}(a, 0), \text{enacting}(b, 0), 2 \rangle \\
(S_6 S_6)^2 & \rightarrow \langle \text{completes}(a), \text{completes}(b), 2 \rangle \\
(S_3 S_3)^2 & \rightarrow \langle \text{enables}(a, g2), \text{enables}(b, g2), 2 \rangle \\
(S_4) & \rightarrow \langle \text{begins}(g2), 2 \rangle \\
& \rightarrow^* \langle \text{completes}(\text{end}), 2 \rangle
\end{align*}
\]

% duration(g1,0) 
% 2 in [1,2] for a 
% 2 in [2,3] for b
Weak Controllability

- Assume:
  - some tasks are *controllable* (e.g., internal to the organization)
  - some tasks are *uncontrollable* (e.g., external to the organization)

- Weak Controllability: *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.

\[
3 \leq T_{total} \leq 7
\]

a solution: 

\[
\text{if } D_{pur} = 1 \text{ then } D_{cc} = D_{col} = 2 \text{ else } D_{cc} = D_{col} = 1
\]
Strong Controllability

Weak Controllability 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.

![Diagram](attachment:image.png)

constraint: \(4 \leq T_{\text{total}} \leq 7\)
a solution: \(1 \leq D_{\text{pack}} \leq 2\)
Solving Controllability Problems using Constrained Horn Clauses (CHCs)

- Constrained Horn Clauses: $H \leftarrow c, A_1, \ldots, A_n$
- 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:

\[
\begin{align*}
\text{begins}(x) & \quad \rightarrow \quad \text{enacting}(x,d) \\
\langle F, t \rangle & \quad \rightarrow \quad \langle F', t \rangle
\end{align*}
\]
(1) Encoding of the semantics

Instantaneous transition:

\[ \langle F, t \rangle \rightarrow \langle F', t \rangle \]

\[ \text{begins}(x) \quad \text{enacting}(x, d) \]

\[ (S_1) \quad \frac{\text{begins}(x) \in F \quad \text{duration}(x, d)}{\langle F, t \rangle \rightarrow \langle (F \setminus \{\text{begins}(x)\}) \cup \{\text{enacting}(x, d)\}, t \rangle} \]
(1) Encoding of the semantics

Instantaneous transition:

\[ <F,t> \rightarrow <F',t> \]

\[ \text{begins}(x) \quad \rightarrow \quad \text{enacting}(x,d) \]

\[
(S_1) \quad \frac{\text{begins}(x) \in F \quad \text{duration}(x,d)}{\langle F, t \rangle \rightarrow \langle (F \setminus \{\text{begins}(x)\}) \cup \{\text{enacting}(x,d)\}, t \rangle}
\]

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

where \( U,C \) are tuples of uncontrollable and controllable durations, resp.
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`

\[ R1: \quad \text{reach}(S,S,U,C) \leftarrow \]
\[ R2: \quad \text{reach}(S0,S2,U,C) \leftarrow tr(S0,S1,U,C), \text{reach}(S1,S2,U,C) \]
(2) Encoding Reachability

- **Reachability Property.**
  
  \[ \text{RP} : \ reachProp(U,C) \leftarrow c(T,U,C), \ reach(init, fin(T), U, C) \]
  
  where \( c(T,U,C) \) is a constraint

- **Initial state.** \( \text{init} : \ < \{\text{begins(start)}\}, 0 > \)

- **Final state.** \( \text{fin}(T) : \ < \{\text{completes(end)}\}, T > \)
Let $\text{Sem}$ 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**

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

  where $\text{adm}(U)$ iff the durations in $U$ belong to the given intervals

- **Strong Controllability**

  $\text{Sem} \cup \{\text{RP}\} \cup LIA \models \exists C. \forall U. \text{adm}(U) \rightarrow \text{reachProp}(U,C)$
(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 ∃−∀ and ∀−∃
  - solvers and CLP systems have termination problems due to recursive reach.
Constrained Horn Clauses: $H \leftarrow c, A_1, \ldots, A_n$

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/\texttt{findall} and derive equisatisfiable \textit{function-free, linear-recursive} clauses:

\[
p(X) \leftarrow c, \ q(Y)
\]

where \( X, Y \) are tuples of variables and \( c \) is a constraint in \textit{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 $RP$

- $Sem \cup \{RP\} \xrightarrow{RI} I_{SP}$ 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)

event(start) ← task(a1) ← par_branch(g1) ← ... 
seq(start, g1) ← seq(g1, g2) ← seq(g1, b) ← ...

uncontrollable(a1) ← controllable(a2) ← controllable(b) ← 
duration(g1, D) ← D = 0 
duration(a1, D) ← 2 ≤ D ≤ 4 
duration(a2, D) ← 1 ≤ D ≤ 2 
duration(b, D) ← 5 ≤ D ≤ 6 

RP: reachProp(A1,(A2,B)) ← true, reach(init,fin(T), A1,(A2,B))

Weak Controllability: ∀ A1. 2 ≤ A1≤ 4 → ∃ A2,B. reachProp(A1,(A2,B))
Removal of the Interpreter (example)

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

  \[
  \text{reachProp}(A1, (A2,B)) \leftarrow A=A1, B=B1, A1 \geq 2, A1 \leq 4, B \geq 5, B \leq 6, \\
  \text{new2}(A, B1, F, G, A1, A2, B)
  \]

  \[
  \text{new2}(A,B1,C,D,A1,A2,B) \leftarrow H=A+C, I=B1-A, J=0, A \geq 1, I \geq 0, A+I \geq 1, \\
  \text{new2}(J, I, H, D, A1, A2, B)
  \]

  \[
  \text{new2}(A,B1,C,D,A1,A2,B) \leftarrow H=B1+C, I=A-B1, J=0, A \geq 1, I \geq 0, A-I \geq 1, \\
  \text{new2}(I,J,H,D,A1,A2,B)
  \]

  \[
  \text{new2}(A,B1,C,D,A1,A2,B) \leftarrow H=A2, A=0, H \geq 1, H \leq 2, \\
  \text{new5}(H,B1,C,D,A1,A2,B)
  \]

  \[
  \text{new5}(A,B1,C,C,A1,A2,B) \leftarrow A=0, B1=0
  \]

  \[
  \text{new5}(A,B1,C,D,A1,A2,B) \leftarrow H=A+C, I=B1-A, J=0, A \geq 1, I \geq 0, A+I \geq 1, \\
  \text{new5}(J,I,H,D,A1,A2,B)
  \]

  \[
  \text{new5}(A,B1,C,D,A1,A2,B) \leftarrow H=B1+C, I=A-B1, J=0, A \geq 1, I \geq 0, A-I \geq 1, \\
  \text{new5}(I,J,H,D,A1,A2,B)
  \]

  \[
  \text{new5}(A,B1,C,D,A1,A2,B) \leftarrow H=A1, A=0, H \geq 2, H \leq 4, \\
  \text{new2}(H,B1,C,D,A1,A2,B)
  \]

- 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.
(1) Generate a disjunction \( a(U,C) \) of constraints

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

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

  \( \text{SOLVE}(I_{SP}, Q) \) returns

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

  - \( false \), otherwise

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

\[
I_{SP} \cup LIA \models \forall U, C. (a(U,C) \rightarrow \text{reachProp}(U,C))
\]
Weak Controllability Algorithm

\[ I_{SP} : \quad q(X) \leftarrow r(X) \]
\[ r(X) \leftarrow X > 0 \]

\[ \text{SOLVE}(I_{SP}, q(X)) \text{ returns the constraint } X > 0 \]

Indeed, \( I_{SP} \cup LIA \models \forall X \quad (X > 0 \rightarrow q(X)) \)
a(U,C) := false;
do {
    Q := (reachProp(U,C) \land \forall C. \neg a(U,C));
    if (SOLVE(I_{SP}, Q) = false) return false;
    a(U,C) := a(U,C) \lor SOLVE(I_{SP},Q);
} while (LIA \nexists U. \text{adm}(U) \rightarrow \exists C. a(U,C));
return a(U,C);
(4) Weak Controllability Algorithm

\[ a(U,C) := false; \]
\[ do \{ \]
\[ \quad Q := (\text{reachProp}(U,C) \land \forall C. \neg a(U,C)); \]
\[ \quad \text{if } (\text{SOLVE}(I_{SP}, Q) = false) \text{ return } false; \]
\[ \quad a(U,C) := a(U,C) \lor \text{SOLVE}(I_{SP}, Q); \]
\[ \} \text{ while } (\text{LIA} \not\models \forall U. \text{adm}(U) \rightarrow \exists C. a(U,C)); \]
\[ \text{return } a(U,C); \]
By definition of SOLVE, the do-while constructs a sequence

\(<a_1(U, C), ..., a_n(U, C)\>\) of disjoint constraints such that

\[\forall U, C. \ a_1(U, C) \rightarrow \text{reachProp}(U, C)\]

\[\land \cdots \land\]

\[a_n(U, C) \rightarrow \text{reachProp}(U, C)\]

that is,

\[\forall U, C. \ (a_1(U, C) \lor \cdots \lor a_n(U, C)) \rightarrow \text{reachProp}(U, C)\]
By definition of SOLVE, the do-while constructs a sequence
\(<a_1(U,C),...,a_n(U,C)>\) of disjoint constraints such that
\[\forall U,C. \ a_1(U,C) \rightarrow \text{reachProp}(U,C) \]
\[\land \ldots \land \ a_n(U,C) \rightarrow \text{reachProp}(U,C)\]
that is, \[\forall U,C. \ (a_1(U,C) \lor \ldots \lor a_n(U,C)) \rightarrow \text{reachProp}(U,C)\] (1)

Moreover, at termination of the do-while:
\[\forall U. \text{adm}(U) \rightarrow \exists C. \ (a_1(U,C) \lor \ldots \lor a_n(U,C))\] (2)

From (1) and (2), by transitivity we get:
\[\forall U. \text{adm}(U) \rightarrow \exists C. \text{reachProp}(U,C) \quad \text{(weak controllability)}\]
as desired.
(4) Weak Controllability Algorithm

\[ a(U, C) := \text{false}; \]
\[ \text{do} \{ \]
\[ \quad Q := (\text{reachProp}(U, C) \land \forall C. \neg a(U, C)); \]
\[ \quad \text{if (SOLVE}(I_{SP}, Q) = \text{false}) \text{ return false; } \]
\[ \quad a(U, C) := a(U, C) \lor \text{SOLVE}(I_{SP}, Q); \]
\[ \text{while (LIA} \not\equiv \forall U. \text{adm}(U) \rightarrow \exists C. a(U, C)); \]
\[ \} \]
\[ \text{return } a(U, C); \]

1) \text{SOLVE}(I_{SP}, \text{reachProp}(A1, (A2, B)) \land \neg \text{false})

returns \( a_1(A1, A2, B) \), which is \( B-2 \leq A1 \leq 4 \land A2 = B-A1 \land 5 \leq B \leq 6. \)

\( \text{LIA} \not\equiv \forall A1. 2 \leq A1 \leq 4 \rightarrow \exists A2, B. a_1(A1, A2, B) \)
Weak Controllability Algorithm

1) \text{SOLVE}(I_{SP}, \text{reachProp}(A_1, (A_2, B)) \land \neg \text{false})

returns \ a_1(A_1, A_2, B), \ \text{which is} \ \ B \leq A_1 \leq 4 \land A_2 = B - A_1 \land 5 \leq B \leq 6.

\text{LIA} \not\models \forall A_1. \ 2 \leq A_1 \leq 4 \rightarrow \exists A_2, B. \ a_1(A_1, A_2, B)

2) \text{SOLVE}(I_{SP}, \text{reachProp}(A_1, (A_2, B)) \land \forall A_2, B. \ \neg a_1(A_1, A_2, B))

returns \ a_2(A_1, A_2, B), \ \text{which is} \ A_1 = 2 \land A_2 = 1 \land B = 6.

\text{LIA} \models \forall A_1. \ 2 \leq A_1 \leq 4 \rightarrow \exists A_2, B. \ (a_1(A_1, A_2, B) \lor a_2(A_1, A_2, B))
(4) Strong Controllability Algorithm

\[
\begin{align*}
a(U, C) &:= \text{false} \\
do \{ \\
Q &:= (\text{reachProp}(U, C) \land \neg a(U, C)));
\text{if} \ (\text{SOLVE}(l_{SP}, Q) = \text{false}) \ \text{return} \ \text{false}; \\
a(U, C) &:= a(U, C) \lor \text{SOLVE}(l_{SP}, Q);
\} \ \text{while} \ (\text{LIA} \not\models \exists C. \forall U. \text{adm}(U) \rightarrow a(U, C));
\text{return} \ a(U, C);
\end{align*}
\]
Implementation

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