Verification of Time-Aware Business Processes using Constrained Horn Clauses

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, Roma, Italy
(3) CNR-IASI, Roma, Italy

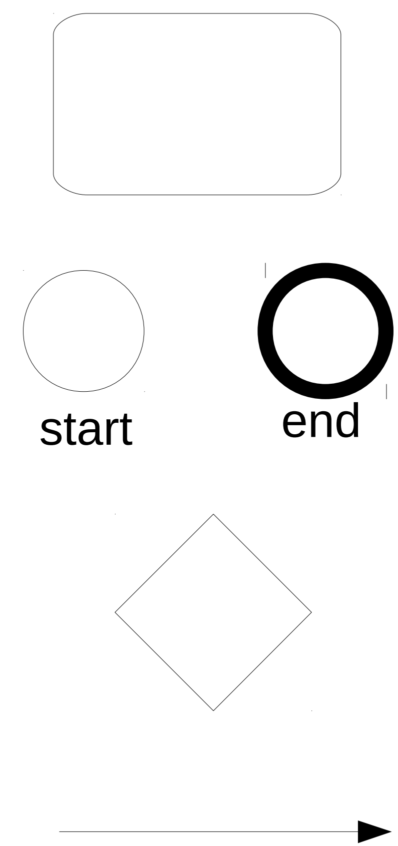
LOPSTR 2016 – Edinburgh (UK)

Talk Outline

- Business Process Model and Notation (BPMN)
- Semantics of time-aware BPMN
- Verification method
  - CHC encoding of the interpreter I
    - BPMN model, semantics, property
  - CHC specialization of I (unfold/fold)
  - Predicate equivalence
  - CHC satisfiability checking using SMT solvers
- Experimental evaluation

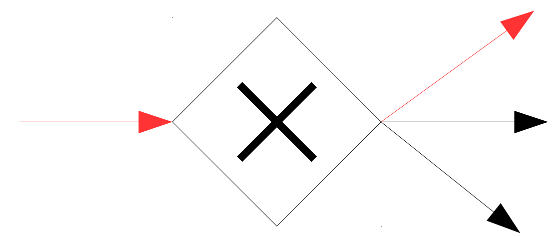
Business Process Modeling and Notation

- Graphical language for modeling organizational processes: activities, events, and their composition (OMG standard)
- Tasks
  - atomic units of work
- Events
  - something that ‘happens’
- Gateways
  - model flow branching / merging
- Sequence flow
  - specifies the order of execution

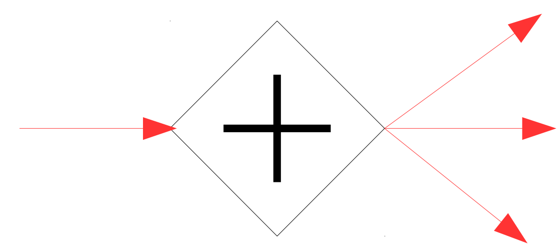


Branch Gateways

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

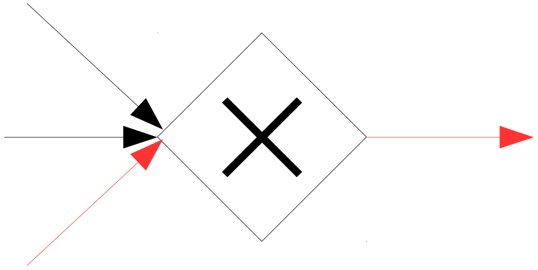


- **parallel** branch gateway (AND)
  - upon activation
    of the incoming flow
    <u>all</u> outgoing flows
    are activated

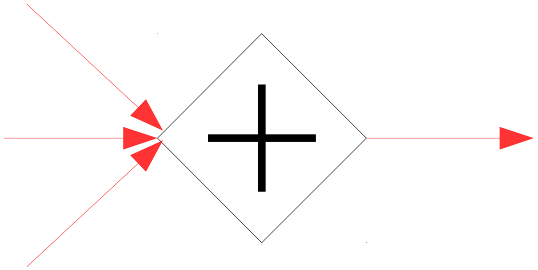


Merge Gateways

- 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

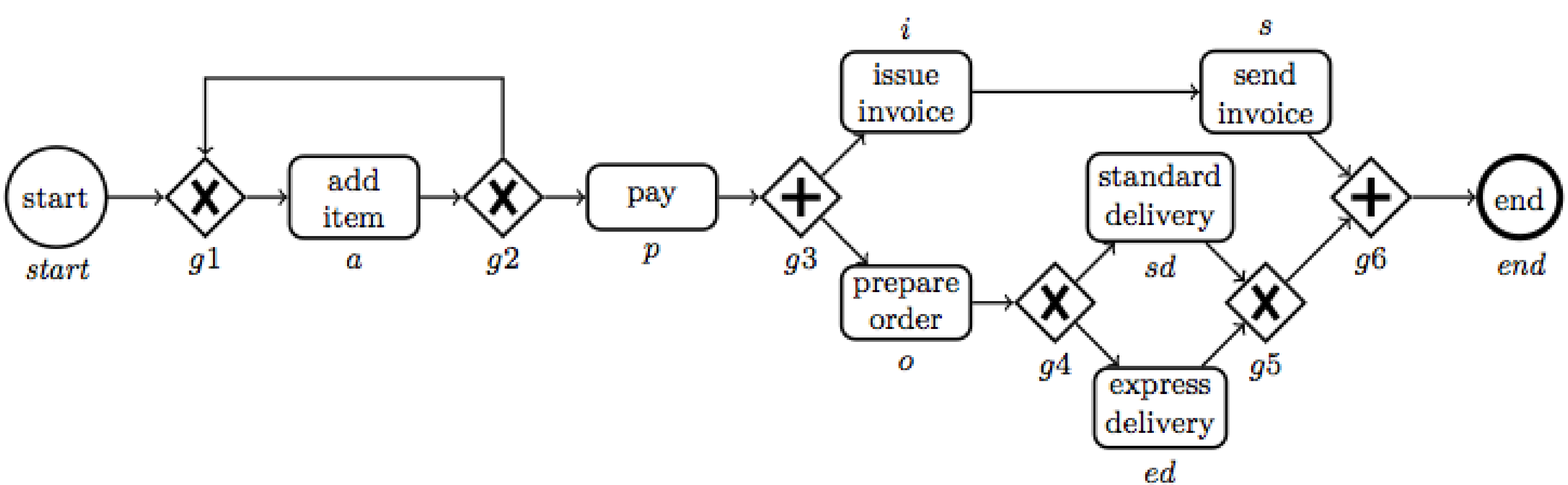


Well-formed BPMN models

- the *start* and *end* events are unique
- all flow objects are on some path from *start* to *end*
- | *Preds*(*start*) | = 0 and | *Succs*(*start*) | = 1
  - similar constraints for the *end* event, tasks and gateways
- on every cyclic path there is at least one occurrence of a task
  - no cycles through gateways only

Purchase Order process (PO)

- A customer adds one or more items to the shopping cart and pays.
- Then, the invoice is sent and the order is delivered.



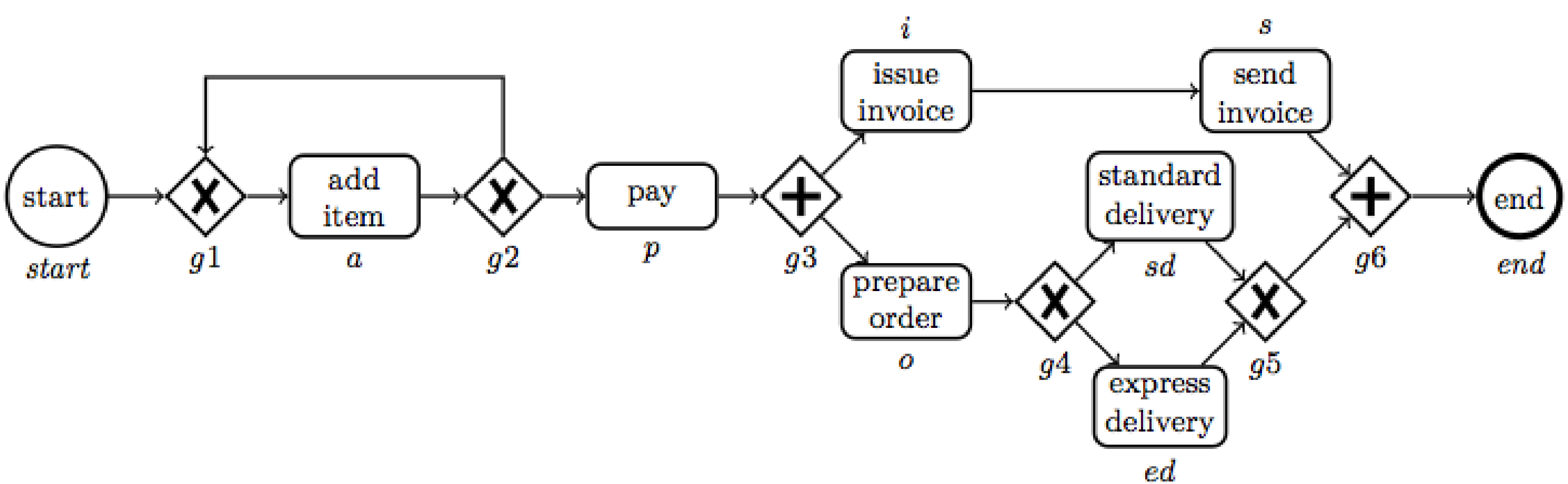
Time-aware BPMN model

- Tasks have a duration
  - a positive integer, with upper and lower bounds
  - for simplicity, events and gateways are instantaneous
- CHC specification
  - $duration(x,d) \leftarrow d_{min} \le d \le d_{max}$.
    x is a task of duration d
  - *task(x).* — x is a task
  - *par_branch(y).* — y is a parallel branch gw
  - *seq*(*x*,*y*). — sequence flow from x to y
- BPMN Meta-model
  - Well-formedness and disjointness of elements

CHC encoding of the Purchase Order process

Events	`start(start). end(end).`
Gateways	`exc_merge(g1).    exc_branch(g2). ...`
Tasks	`task(a).    task(p).  ...`
Sequence flow	`seq(start,g1).  seq(g1,a). ...`
Task durations	see below

```
duration(a, D):- D>=1, D=<6.  % add item
duration(p, D):- D>=1, D=<2.  % pay
...
duration(X, D):- not_task(X), D=0.
                                    % gateways and events
```



Semantics of time-aware BPMN

- Transition relation → between states <*F*,*t*>
- *t* time point 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* has completed its execution
  - *enables*(*x,y*): *x* has completed its execution and enables its successor *y*
    - x,y denote flow objects (tasks, or events, or gateways)

Semantics of time-aware BPMN

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

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

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

Semantics of time-aware BPMN

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

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

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

Semantics of time-aware BPMN

- Time elapses

$$(S_7)\quad \frac{no\_other\_premises(F) \qquad \exists x\, \exists r\ enacting(x,r) \in F \qquad m > 0}{\begin{array}{l}\langle F, t\rangle \longrightarrow \langle (F \setminus \{enacting(x,r) \mid enacting(x,r) \in F\}) \\ \qquad \cup \{enacting(x, r-m) \mid enacting(x,r) \in F\},\ t+m\rangle\end{array}}$$

where: (i) $no\_other\_premises(F)$ holds iff none of the rules S_1–S_6 has its premise true, and (ii) $m = min\{r \mid enacting(x,r) \in F\}$.

CHC Encoding of the Semantics

- The predicate `tr` encodes the operational semantics $\longrightarrow$

```
S1. tr(s(F,T), s(FU,T)):- member(begins(X),F), duration(X,D),
                          update(F,[begins(X)],[enacting(X,D)],FU).


S2. tr(s(F,T), s(FU,T)):- member(completes(X),F), par_branch(X),
                          findall(enables(X,S),(seq(X,S)),Enbls),
                          update(F,[completes(X)],Enbls,FU).


S3. tr(s(F,T), s(FU,T)):- member(completes(X),F),
                          not_par_branch(X),   seq(X,S),
                          update(F,[completes(X)],[enables(X,S)],FU).
...
```

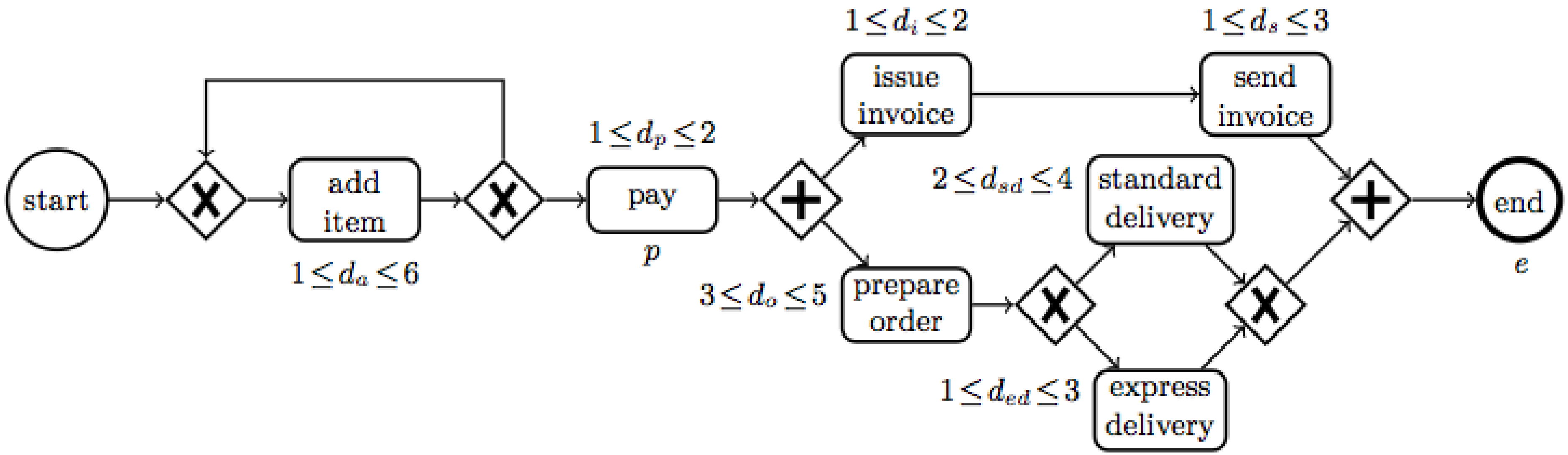
- The predicate `reach` encodes reachability $\longrightarrow^*$

```
R1. reach(S,S).

R2. reach(S,S2) :- tr(S,S1), reach(S1,S2).
```

CHC Encoding of the Semantics

```
S4. tr(s(F,T), s(FU,T)) :- member(enables(_,_),F),   par_merge(X),
                          findall(enables(P,X),(seq(P,X)),Enbls),
                          sublist(Enbls,F),
                          update(F,Enbls,[begins(X)],FU).


S5. tr(s(F,T), s(FU,T)):- member(enables(P,X),F),   not_par_merge(X),
                          update(F,[enables(P,X)],[begins(X)],FU).


S6. tr(s(F,T), s(FU,T)):- member(enacting(X,R),F), R=0,
                          update(F,[enacting(X,R)],[completes(X)],FU).


S7. tr(s(F,T), s(FU,TU))  :- 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),
                   update(F,Enacts,EnactsU,FU), TU=T+M.
```

Verification of the PO process



- Property to be verified

prop: if $init \to^* \langle\{completes(p)\}, t_p\rangle \to^* \langle\{completes(e)\}, t_e\rangle$, then $t_e \leq t_p + 9$.

- CHC encoding of the (negation of the) property

```
NP.     false :-    Ts=0, Te>Tp+9,
                    reach(s([begins(start)],Ts), s([completes(p)],Tp)),
                    reach(s([completes(p) ],Tp), s([completes(e)],Te)).
```

- The property holds iff the set I = {S1-S7,R1-R2,PO,NP} of clauses is satisfiable.

Program transformation

- Cannot check satisfiability of I using SMT solvers
  - lists, terms, findall predicate
- CLP systems do not terminate
  - recursive reach predicate
- Removal of the interpreter
  - Rule-based transformation strategies (unfolding, definition, folding, useless/subsumed clauses removal)
  - Enables the use of CHC/SMT solvers
- Predicate equivalence
  - Possibly reduces the number of predicates

Removal of the interpreter

- From I derive an equisatisfiable set of clauses I_{sp}
  - I satisfiable iff I_{sp} satisfiable
- I_{sp} contains
  - no references to the CHC encoding of the business process and of the semantics
  - no lists, no terms, no `findall`
  - Clauses of the form

```
new21(A,B,C) :- A=0, D=<3, D>=1, new10(D,B,C).
```

- We can can apply CHC/SMT solvers for checking the satisfiability of I_{sp}

Removal of the Interpreter

```
false :-  A=0, B>=1, B=<2, C>=1, C=<6,
          D>=3, D=<5, E>0, F-E>9,
          new1(C,A,E), new2(B,D,E,F).

new1(A,B,C) :- A=0, D=<6, D>=1, new1(D,B,C).
new1(A,B,C) :- A=0, D=<2, D>=1, new44(D,B,C).
new1(A,B,C) :- D=0, E=A+B, A>0,  new1(D,E,C).

new44(A,B,C) :- A=0, B=C.
new44(A,B,C) :- D=0, E=A+B, A>0, new44(D,E,C).

...
```

Predicate Equivalence

- Decidable predicate equivalence test
  - based on predicate renaming and constraint equivalence
- Discovers classes of equivalent predicates
  - `{{new17, new11},{new6, new7}}`

```
new17(A,B,C) :- A=0, B=C.
new17(A,B,C) :- D=0, E=A+B, A>0, new17(D,E,C).
new6(A,B,C,D) :- B=0, new17(A,C,D).
```

```
new11(A,B,C) :- A=0, B=C.
new11(A,B,C) :- D=0, E=A+B, A>0, new11(D,E,C).
new7(A,B,C,D) :- B=0, new11(A,C,D).
```

Experimental evaluation

Process / Property	RI	Preds	Z3	Answer		PE	Preds %	Z3 %
1. PO								
Q1.1	0.42	14	0.87	*true*		0.00	64%	68%
Q1.2	0.23	14	0.75	*true*		0.01	71%	71%
Q1.3	0.26	5	0.10	*false*		0.00	100%	100%
2. RDOA								
Q1.1	0.07	9	0.31	*false*		0.00	78%	74%
3. STEMI_ED								
Q3.1	0.25	12	1.05	*true*		0.00	75%	91%
Q3.2	0.27	4	0.09	*false*		0.00	100%	100%
Q3.3	0.29	19	1.80	*true*		0.00	79%	91%
4. STEMI_ED_CCU								
Q4.1	1.39	36	11.11	*true*		0.03	75%	87%
Q4.2	0.10	14	34.21	*true*		0.00	79%	37%
Q4.3	0.06	8	2.11	*false*		0.00	100%	100%

- Similar results using EldaRICA

Conclusions

- Flexible framework for reasoning about BP
  - Parametric w.r.t. the semantics and property
  - Satisfiability-preserving program transformations
    - enables the use of state-of-the-art SMT solvers

- Future developments
  - different semantics of time, different properties
  - data (Montali, Deutsch, ...)
  - ontologies (Proietti&Smith)
  - VeriMAP system
    - http://www.map.uniroma2.it/VeriMAP/

The end

Thank you!

1. PO

- % Q1.1 (paper running example)

  % after payment is completed, the process takes at most 9 time units to complete

  r_clause(incorrect1, [reachable(s([begins(start)],T1),s([completes(p)],T2)), T1=0, reachable(s([completes(p)],T2), s([completes(end)],T3)), T2>T1, T3-T2>9]).

- % Q1.2 the process takes at least 4 time units to complete

  r_clause(incorrect2, [reachable(s([begins(start)],T1),s([completes(end)],T2)), T2-T1=<3]).

- % Q1.3 - FALSE
  % the task prepare order (o) and the task send invoice (s) cannot be enacting simultaneously

  r_clause(incorrect3, [reachable(s([begins(start)],T1),
  s([enacting(o,O),enacting(s,S)],T2)), S>=1, O>=1]).

2. RDOA

- Request Day-Off Approval
  - Huai et al. Towards Trustworthy Composite Service Through Business Process Model Verification UIC-ATC 2010

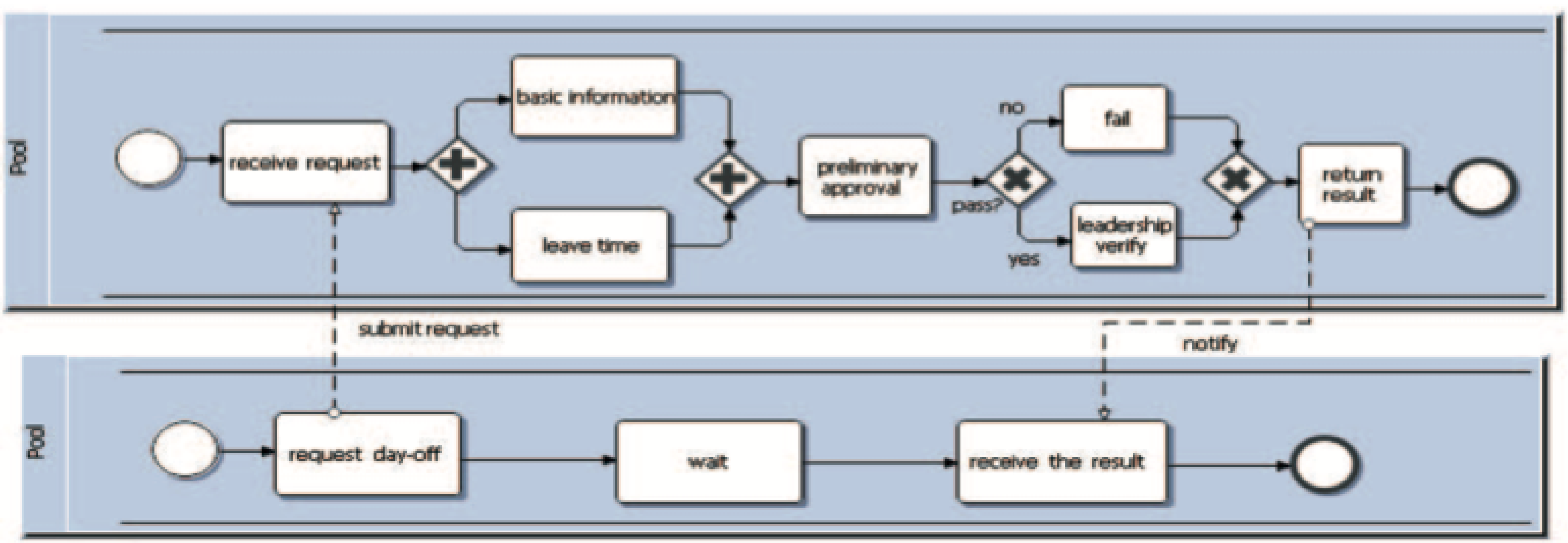


Figure 1. Request day off approval process

- % Q2.1 - FALSE
- % After the employee submits the day-off request, a result is returned in less than 120 time units.
- r_clause(incorrect1, [reachable(s([begins(s)],T1),s([completes(return_result)],T2)), T2-T1>=120]).

3. STEMI: ST-segment Evaluation Myocardial Infarction

- ED (Emergency Department) Admission
  - Controllability in temporal Conceptual workflow Schemata Combi et. al. BPM 2009

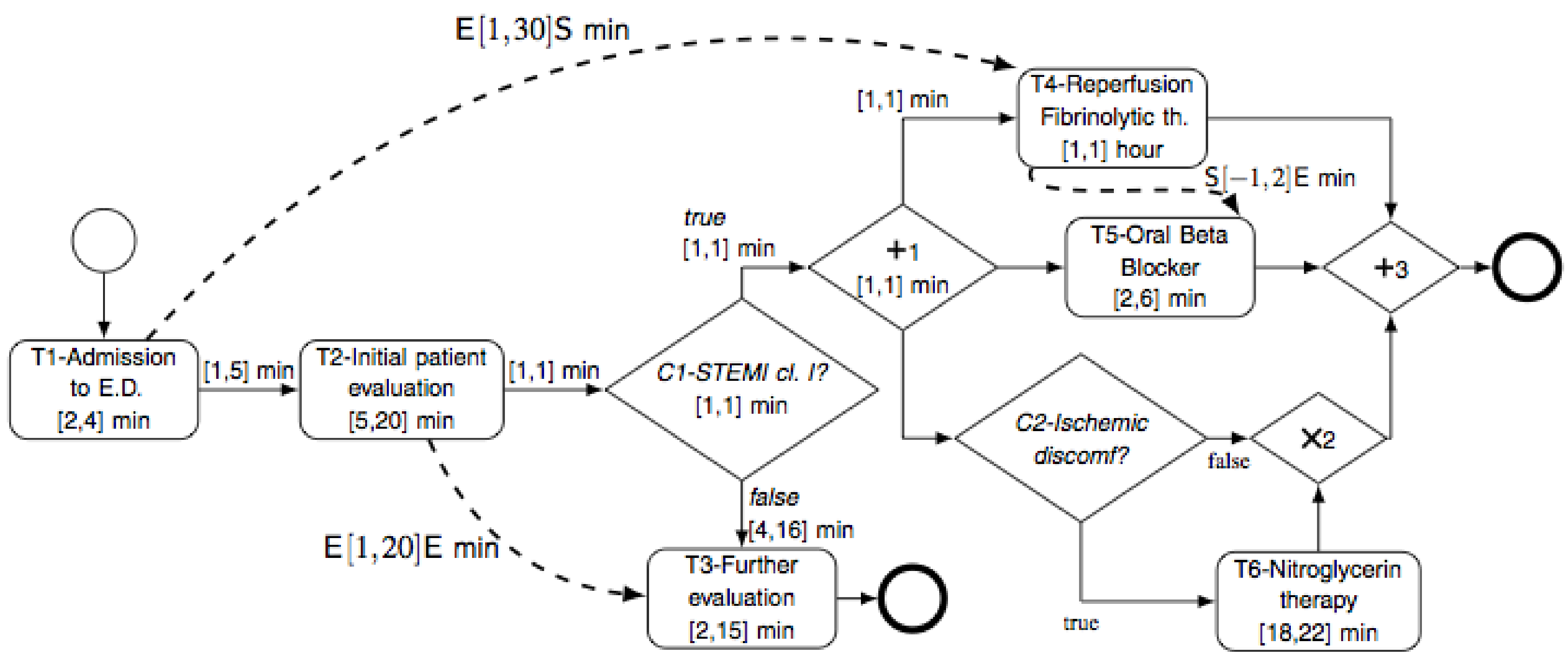


Fig. 6. Workflow graph example of patient admission to an hospital. The dashed edges represent relative constraints.

- % Q3.1

  % since the completion of g1, the process takes at most 22 minutes to be completed

  r_clause(incorrect1, [reachable(s([completes(g1)],T1),s([completes(g3)],T2)), T2-T1>=23]).

- % Q3.2 - FALSE

  % since the completion of T2, it takes at most 20 minutes to begin T3

  r_clause(incorrect2, [reachable(s([completes(t2)],T1),s([completes(t3)],T2)), T2-T1>=21]).

- % Q3.3 % the process takes at least 11 minutes to be completed

  r_clause(incorrect3, [reachable(s([begins(start)],T1),s([completes(end)],T2)), T2-T1=< 10]).

4. STEMI-CCU: ST-segment Evaluation Myocardial Infarction

- ED (Emergency Department) Admission + CCU (Coronary Care Unit) Admission
  - Combi et al.. Conceptual Modeling of Flexible Temporal Workflows. ACM AAS 2012

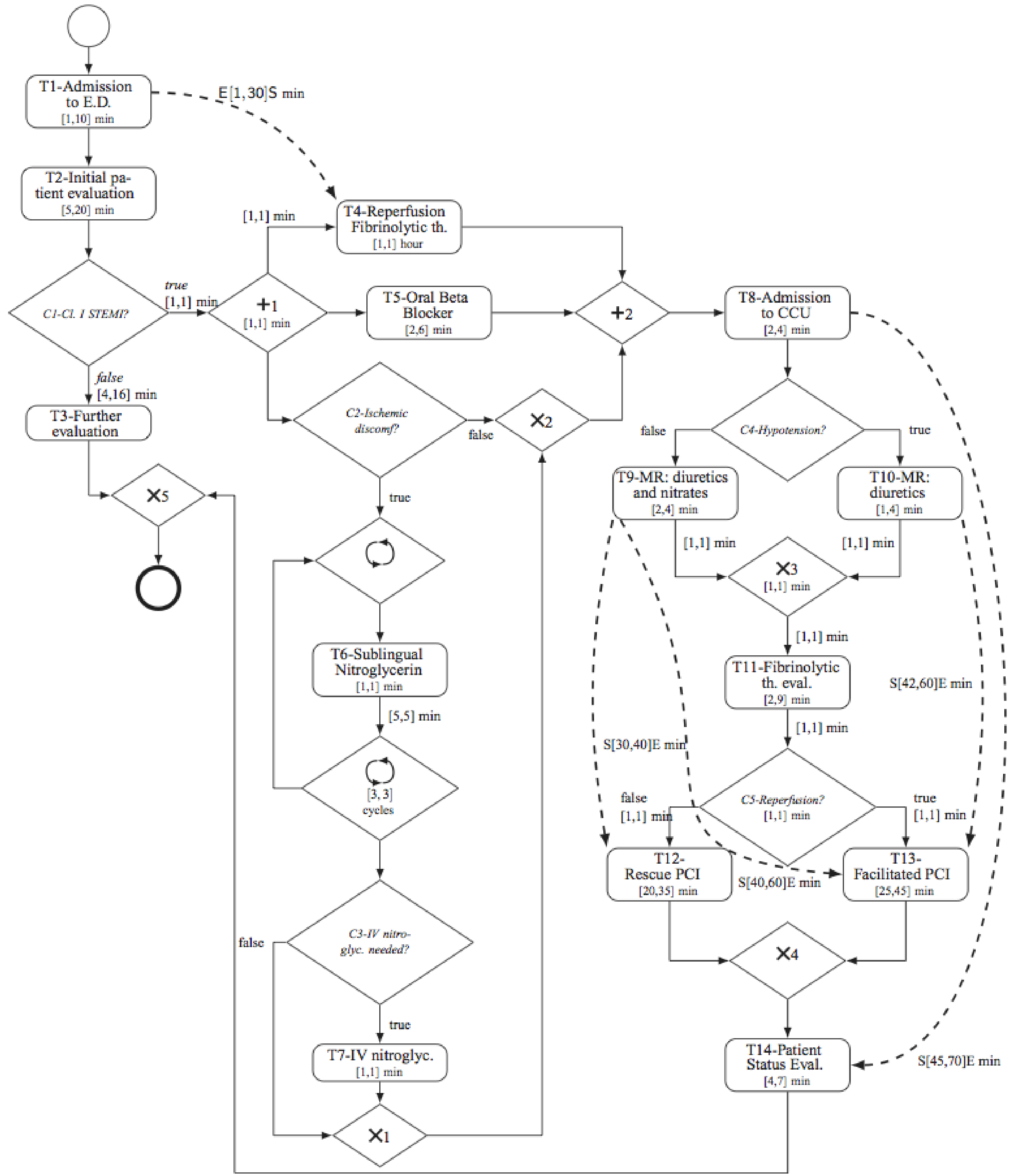


Fig. 5. Workflow graph example of patient admission to an hospital. The dashed edges represent relative constraints. It is worth noting that the last Or-join ×5 could be avoided, maintaining the same expressivity, if we allow more Ends, one for each alternative path.

- % Q4.1 % since the beginning of the admission to E.D. procedure, it takes at most 96 minutes for a patient to be admitted to the CCU

  r_clause(incorrect1, [reachable(s([begins(t1)],T1),s([completes(t8)],T2)), T2-T1>=97]).

- % Q4.2 % since the beginning of T8, it takes at least 35 minutes to complete T4

  % Therefore, it satisfies (the lowerbound of) the constraint S_T8[45,70]E_T14

  r_clause(incorrect2, [reachable(s([begins(t8)],T1),s([completes(t14)],T2)), T2-T1=<34]).

- % Q4.3 - FALSE

  % since the beginning of T10, it takes at most 60 minutes to complete T13

  % Therefore, it violates (the upperbound of) the constraint S_T10[42,60]E_T13

  r_clause(incorrect3, [reachable(s([begins(t10)],T1),s([completes(t13)],T2)), T2-T1>=61]).

Unfold/Fold program specialization

incorrect :- initConf(C),reach(C,C1),errorConf(C1).

- **UNFOLDING** (replace initConf(C) with the body of its definition)

  incorrect :- X>=1,Y>=1, reach(cf(cmd(3,ite(neq(x,y)),4,h), [(x,X),(y,Y)],[]), C1),
  errorConf(C1).

- **UNFOLDING** (wrt errorConf(C1))

  incorrect :- X>=1,Y>=1, X1=<-1, reach(cf(cmd(3,ite(neq(x,y)),4,h),[(x,X),(y,Y)],[]),
  cf(cmd(h,halt),[(x,X1),(y,Y1)],[]))).

- **DEFINITION-INTRODUCTION** (with constraint generalization)

  new3(X,Y, X1,Y1) :- reach(cf(cmd(3,ite(neq(x,y)),4,h),[(x,X),(y,Y)],[]),
  cf(cmd(h,halt),[(x,X1),(y,Y1)],[]))).

- **FOLDING** (replace an instance of the body of a definition by its head)

  incorrect :- X>=1,Y>=1,X1=<-1, new3(X,Y, X1,Y1).