Supervision of Constraint-based Processes: a Declarative Perspective

Sauro Schaidt, Eduardo A. P. Santos,
Agnelo Denis Vieira, Eduardo de F. R. Loures

Pontifical Catholic University of Parana
Graduate Program in Industrial and Systems Engineering
Graduate Program in Health Technology
Curitiba, Brazil

SCIENTIA, VITA ET FIDES

PUCPR

2d International Workshop on Adaptive Case Management and other non-workflow approaches to BPM (ACM'13)

Outline

- Introduction
- Overview
- Supervisory Control Theory (SCT)
- Supervision of constraint-based processes
- Application example

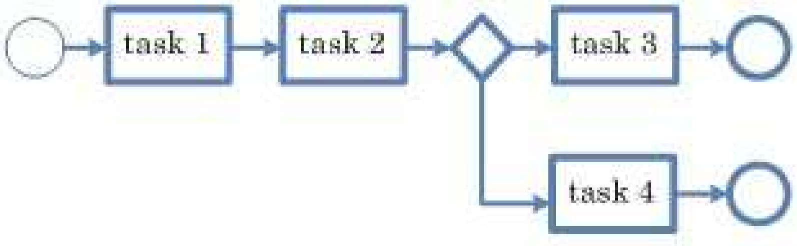
Introduction

Constraint-based processes

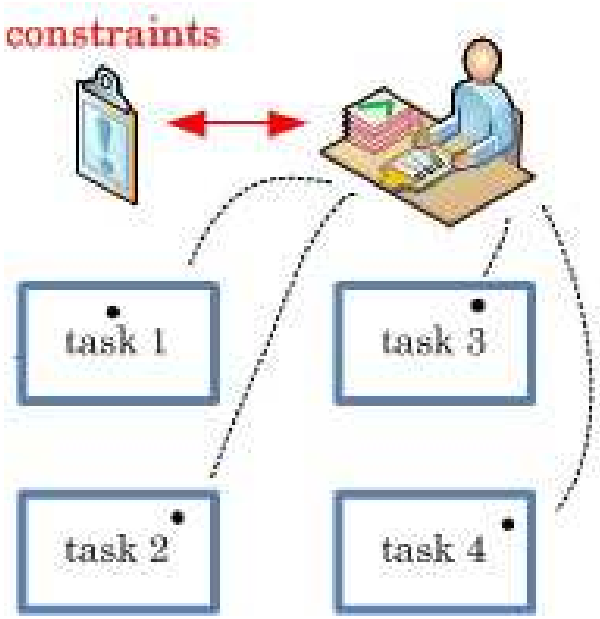
- In this kind of process it is difficult to envision all possible paths and the process is driven by user decisions rather than by system decisions
- Constraint: *at least one of the four tasks has to be executed, but all of them can be executed and each of them may be executed an arbitrary number of times*

Introduction

- Constraint-based processes are less repetitive and the emphasis is on flexibility and user empowerment



(a) imperative



(b) declarative

Overview

Constraint-based processes meet control theory

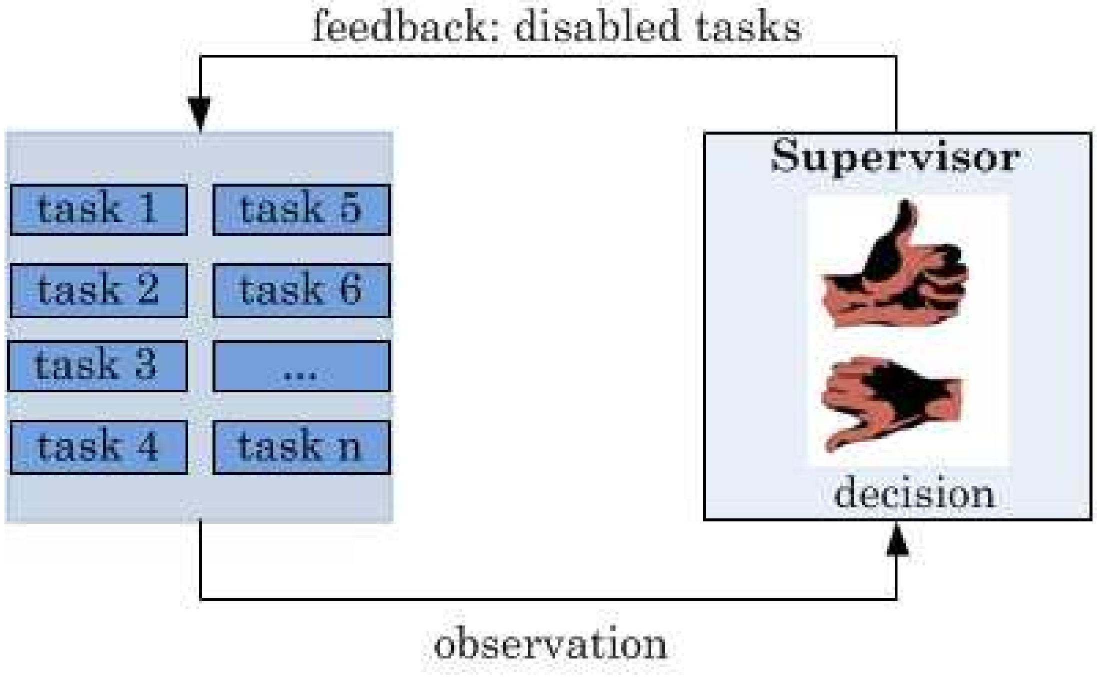
- Aim of our proposal: to build a supervisor (with some control logic) that restrain a process to a particular behavior

- The control action of supervisor is to give a list of disabled (or enabled) tasks on possible next steps

- Supervision is permissive: It does not force but just enables or disables tasks

Overview

Constraint-based processes meet control theory



⇩

Goal: Achieve the desired behavior !!

Overview

- Approach based on Supervisory Control Theory or Ramadge and Wonham model

- Basic reference: Ramadge, P. J. and Wonham, W. M., 1989. The control of discrete event systems. In: Proc. of IEEE - Special Issue on Discrete Event Dynamic Systems, 77 (1), 81-98

Supervisory Control Theory (SCT)

- The start point to apply the SCT is to consider that any DES has an underlying event set associated to it (for instance, the start or completion of a task)

- This set can be seen as the “alphabet” of a language and event sequences can be thought of as “words” in that language

- We can pose questions such as “Can we control a DES in such way that it speaks a given language?” or “Which language does this DES must avoid?”

Supervisory Control Theory (SCT)

- So we consider that a DES may contain strings that are not acceptable because they violate some rule or constraint that we wish to impose on the system

- The premise is that this behavior is not satisfactory and must be "modified" by control (possibly to a subset of this behavior)

- In order to alter the behavior of DES we introduce a supervisor

Supervisory Control Theory (SCT)

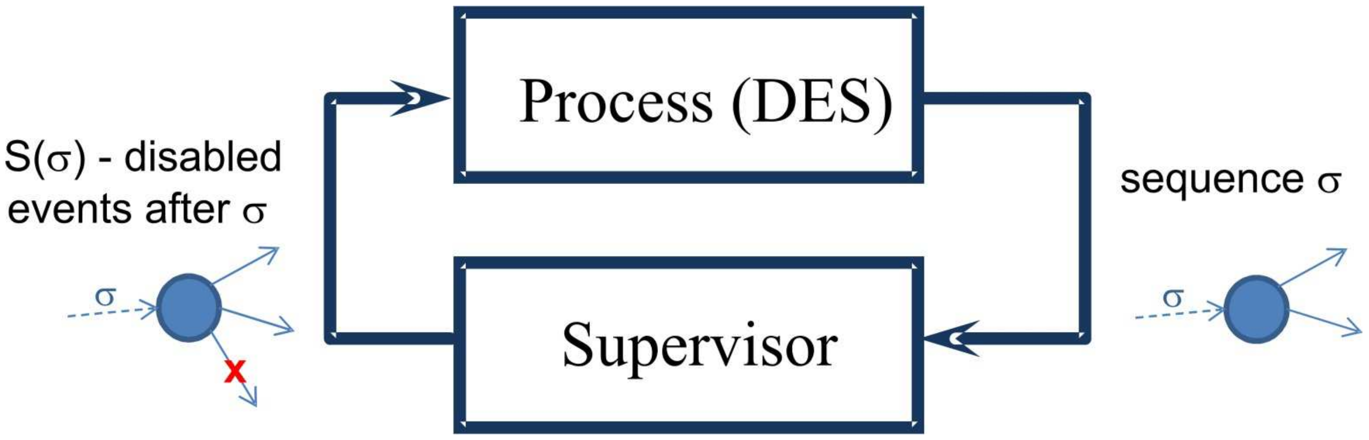
- The Supervisor restrains the DES behavior by dynamically disabling certain events

- *The Supervisor does not force what should happen, but defines what cannot happen!*

Supervisory Control Theory (SCT)

The control

- Some events can be disabled by an external controller: the supervisor
- SCT allows the designer to consider that the occurrence of some events may be avoided by a controller while some others cannot be avoided



Supervisory Control Theory (SCT)

Control structure

Partition of the alphabet Σ: $\Sigma = \Sigma_c \cup \Sigma_u$

Σ_c : set of controllable events that can be disabled

Σ_{uc} : set of uncontrollable events that can not be disabled

Control input:

Subset $\gamma \subseteq \Sigma$ such that

If $\sigma \in \gamma$ it is allowed by S, otherwise σ is disabled by S.

Supervisory Control Theory (SCT)

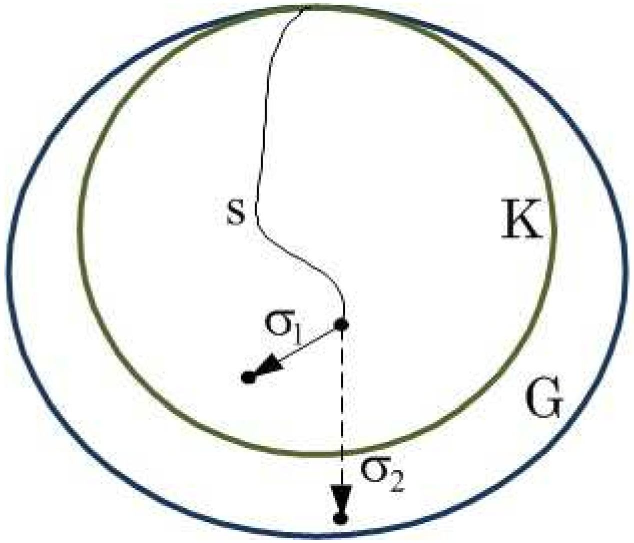
Controlability

- The key existence result for supervisors in the presence of uncontrollable events is the Controllability Theorem
- Formally: $\overline{K}\sigma \cap L(G) \subseteq \overline{K}$
- *Every illegal sequence* σ *can be certainly avoid by the supervisor*

$\sigma_1, \sigma_2 \in \Sigma_{uc}$

$s\sigma_1 \in K \rightarrow OK$

$s\sigma_2 \notin K$ → must be avoided!



Supervision of constraint-based processes

Methodology

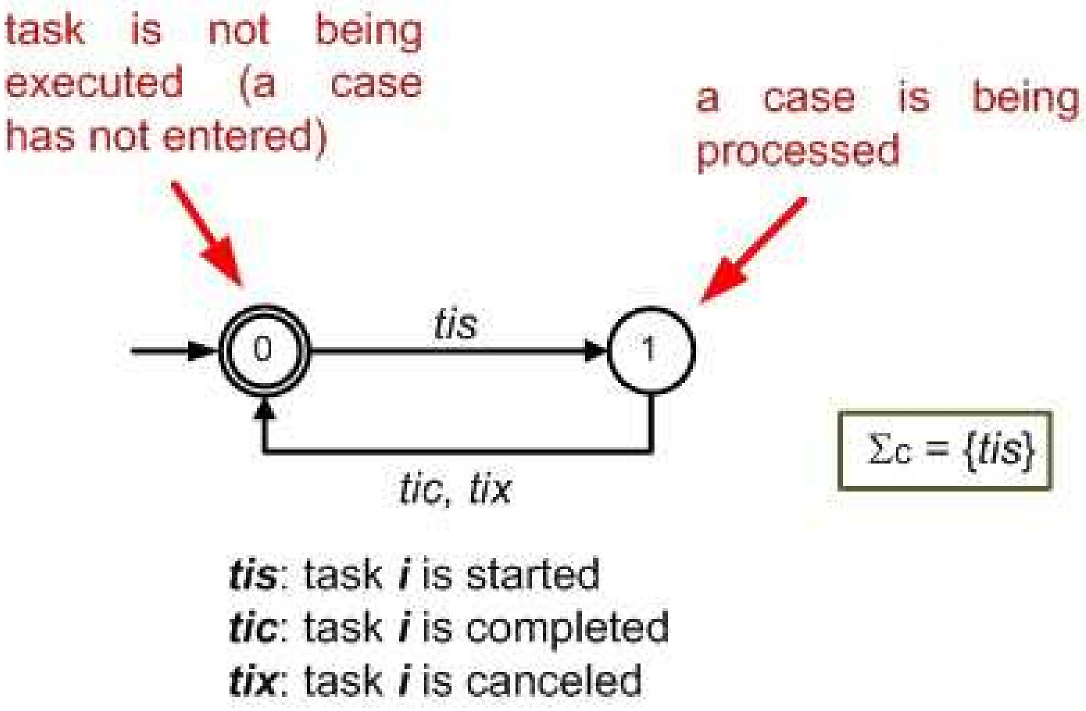
1) Modeling of tasks

2) Modeling of constraints

3) Supervisor synthesis

Supervision of constraint-based processes

1) Modeling of tasks


task is not being executed (a case has not entered)

a case is being processed

$\Sigma c = \{tis\}$

tis: task ***i*** is started
tic: task ***i*** is completed
tix: task ***i*** is canceled

Supervision of constraint-based processes

2) Modeling of constraints

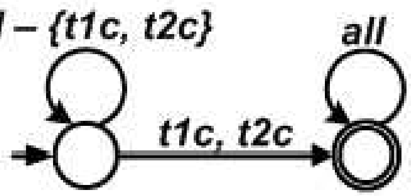
- Set of constrains as proposed by Pesic (2008):

Pesic, M. (2008) “Constraint-based workflow management systems: Shifting control to users”, Phd thesis, Eindhoven University of Technology, Eindhoven.

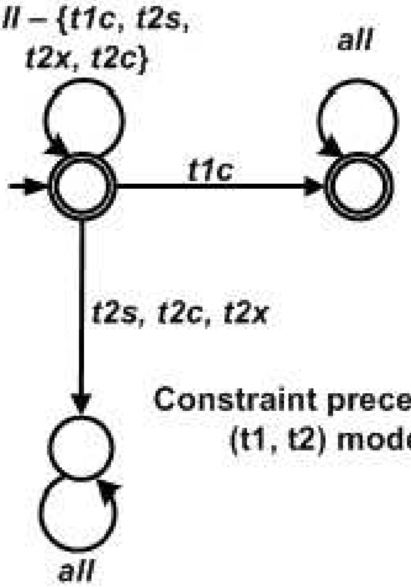
- In our case: automata

Supervision of constraint-based processes

2) Modeling of constraints



Constraint 1of2 (t1, t2) model



Constraint precedence (t1, t2) model

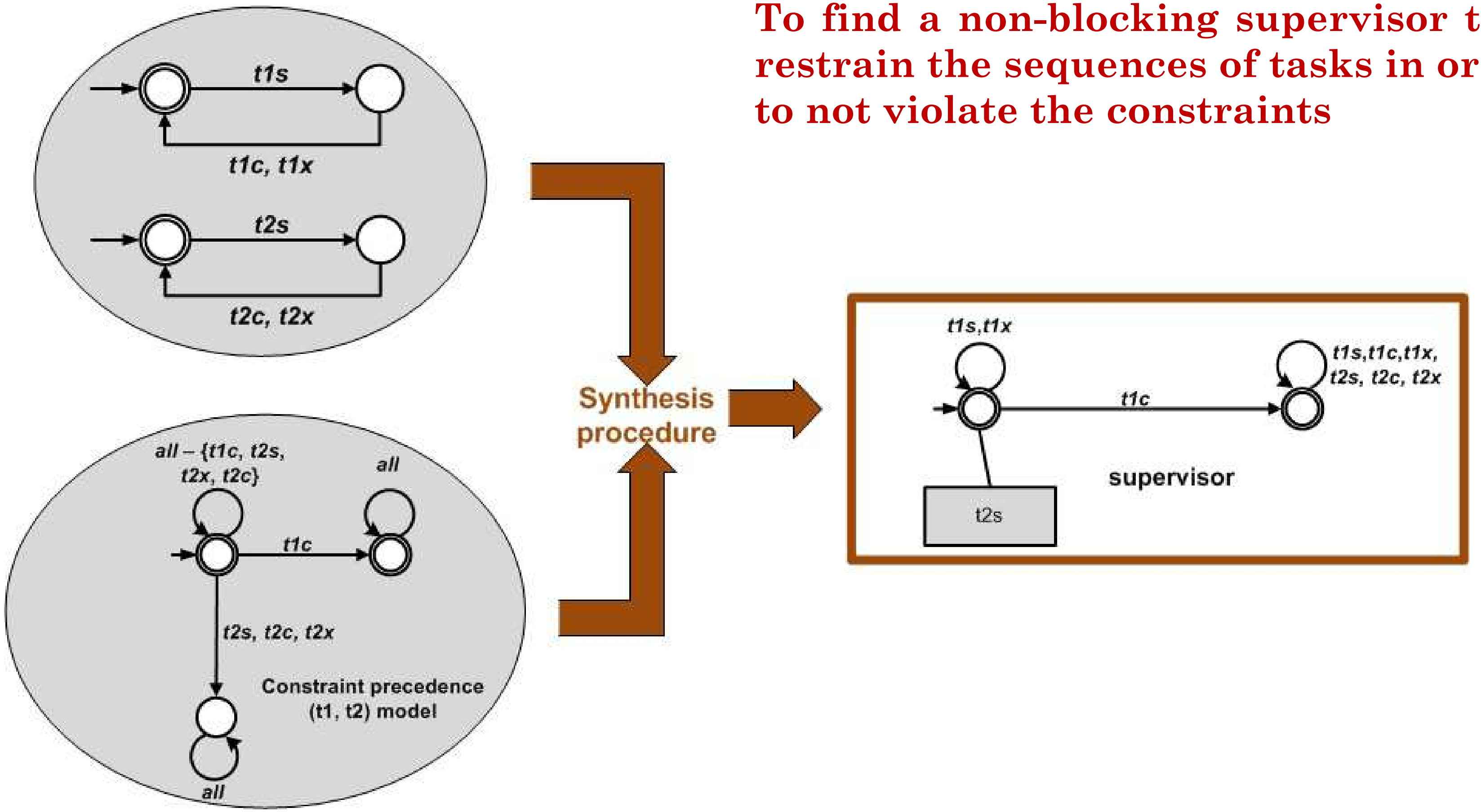
The 1of2 model specifies that at least one of the two tasks t1 and t2 has to be executed, but both can be executed and each of them can be executed an arbitrary number of times.

The precedence model requires that task t2 is preceded by task t1, i.e., it specifies that task t2 can be executed only after task t1 is executed.

Supervision of constraint-based processes

3) Synthesis procedure

To find a non-blocking supervisor that restrain the sequences of tasks in order to not violate the constraints

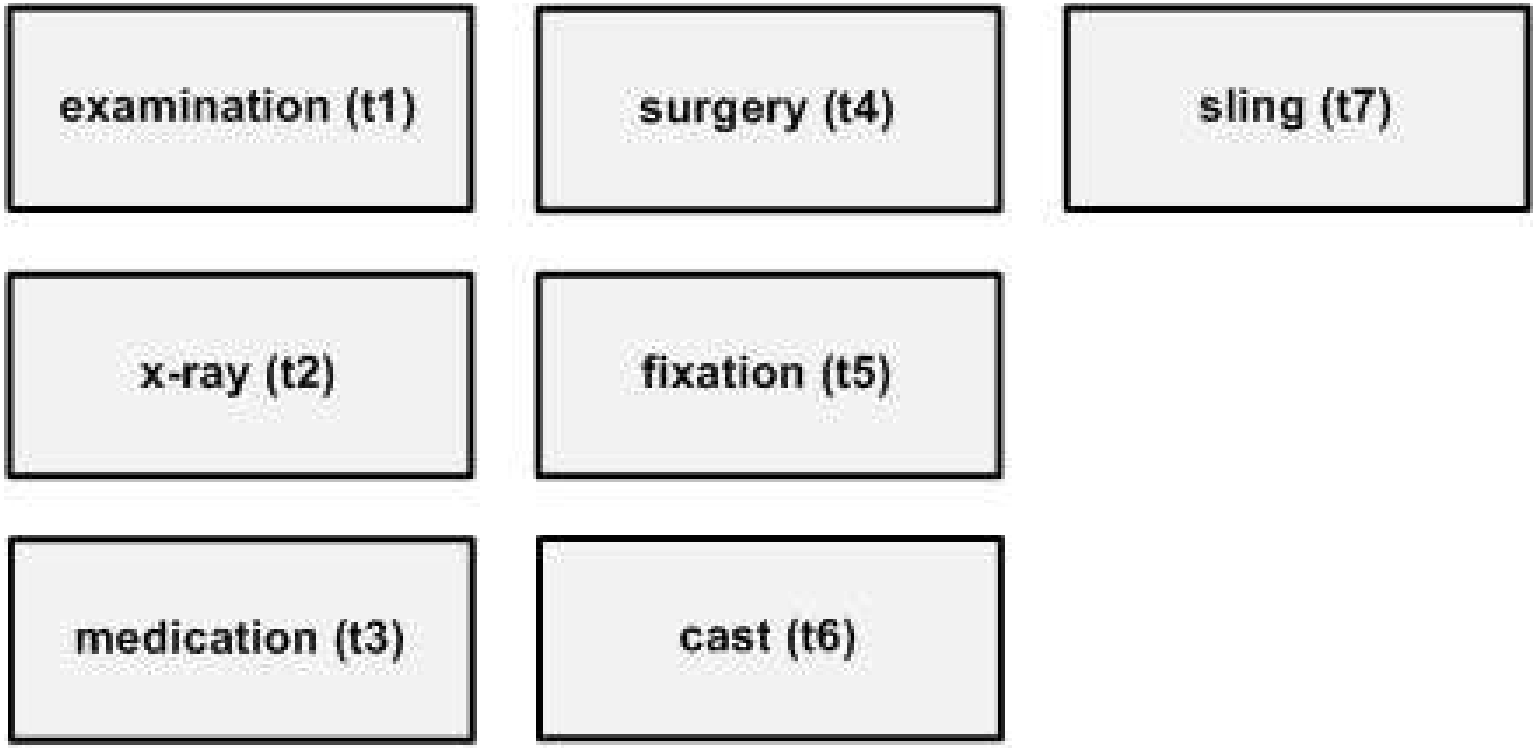

t1s
t1c, t1x
t2s
t2c, t2x
all – {t1c, t2s, t2x, t2c}
all
t1c
t2s, t2c, t2x
Constraint precedence (t1, t2) model
all
Synthesis procedure
t1s,t1x
t1c
t1s,t1c,t1x, t2s, t2c, t2x
supervisor
t2s

Example of application

A process for handling a patient at the first aid department in a hospital with a suspicion of a fracture

Example of application

Tasks:



examination (t1)

surgery (t4)

sling (t7)

x-ray (t2)

fixation (t5)

medication (t3)

cast (t6)

Example of application

Constraints:

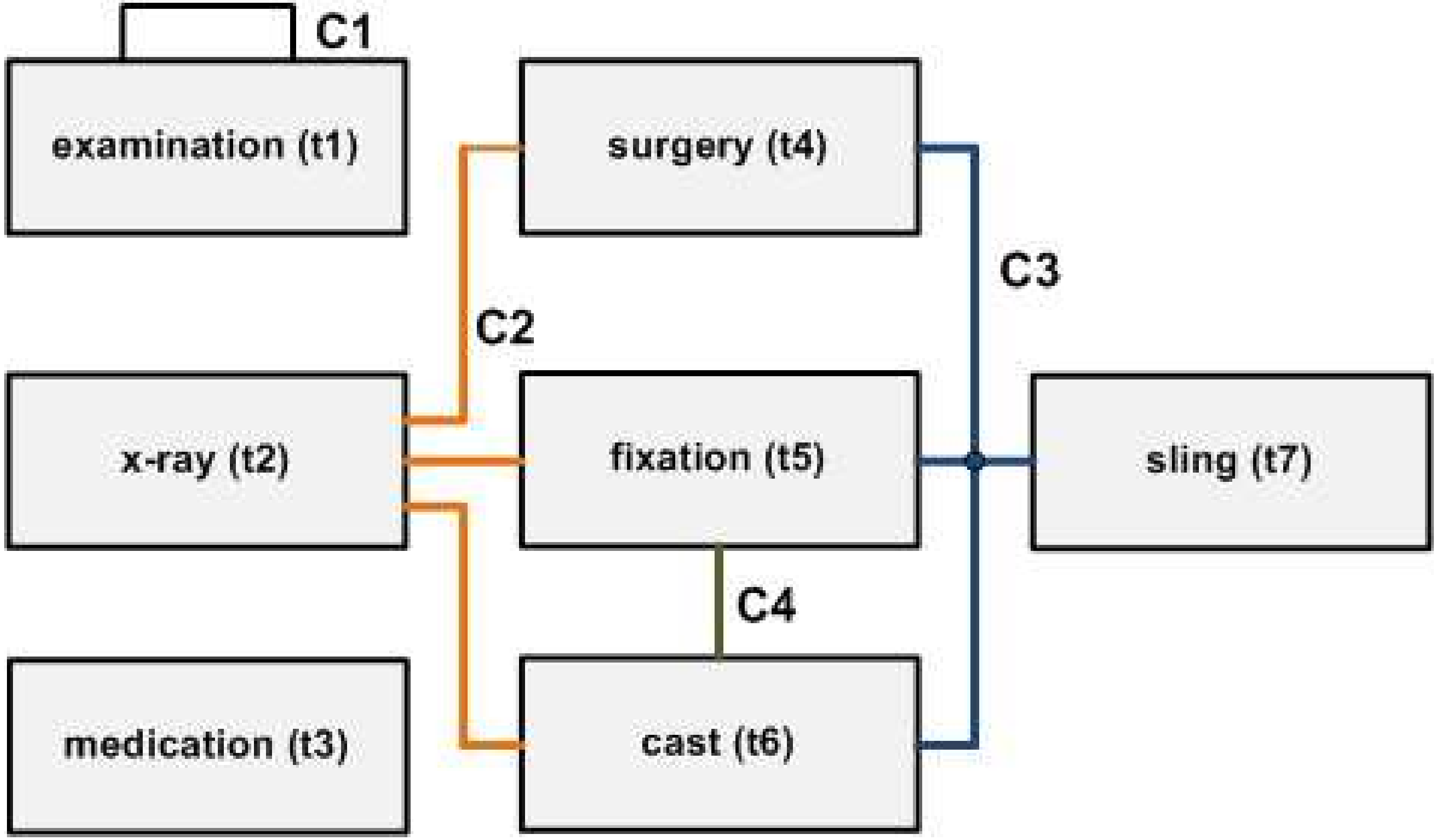
1) Task “examination” (t1) must be the first executed task

2) Tasks “surgery” (t4), “fixation” (t5), and “cast” (t6) must be preceded by task “x-ray”

3) One of the four tasks has to be executed: “surgery” (t4), “fixation” (t5), “cast” (t6), “sling” (t7)

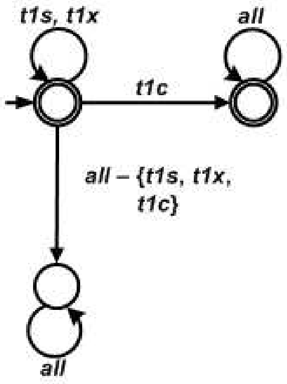
4) If “fixation” (t5) is executed then “cast” (t6) cannot be executed (and vice-versa)

Example of application

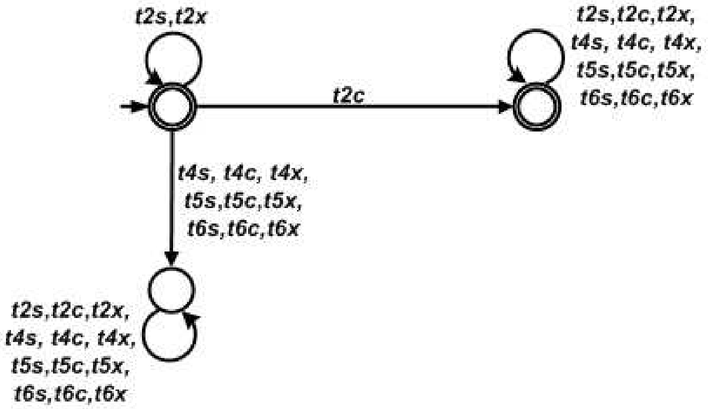


Example of application

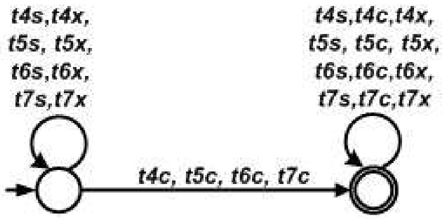
Constraint init (t1) specifies that task examination must be the first executed task in an instance



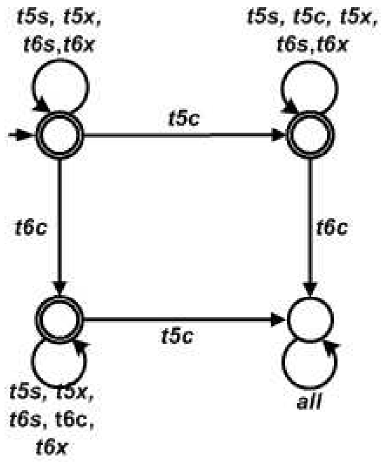
Constraint precedence (t2, (t4,t5,t6)) specifies that surgery, fixation and cast require X-ray to rule out the presence of a fracture, or to decide how to treat the fracture



Constraint 1of4 (t4,t5,t6,t7) specifies that the treatments can be given in any combination and each patient receives at least one treatment

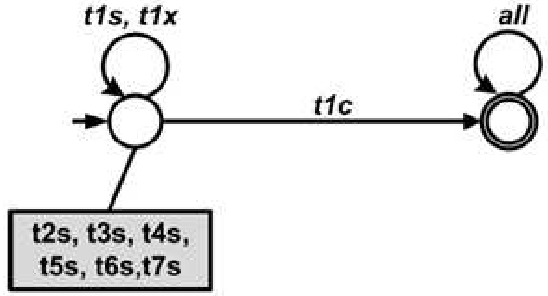


Constraint not coexistence (t5, t6) specifies that cast and fixation are mutually exclusive

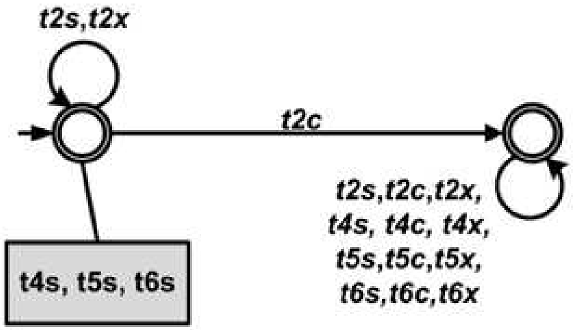


Example of application

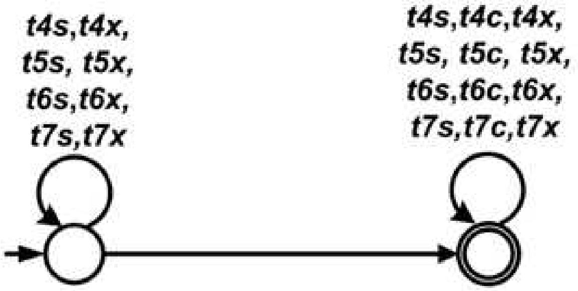
Supervisor (constraint init)



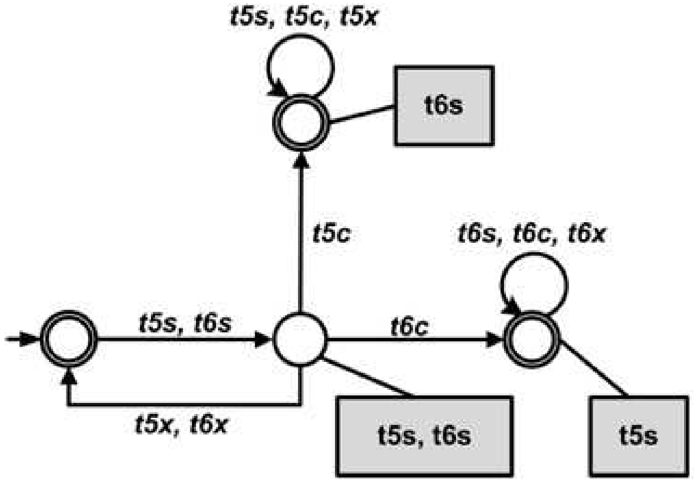
Supervisor (constraint precedence)



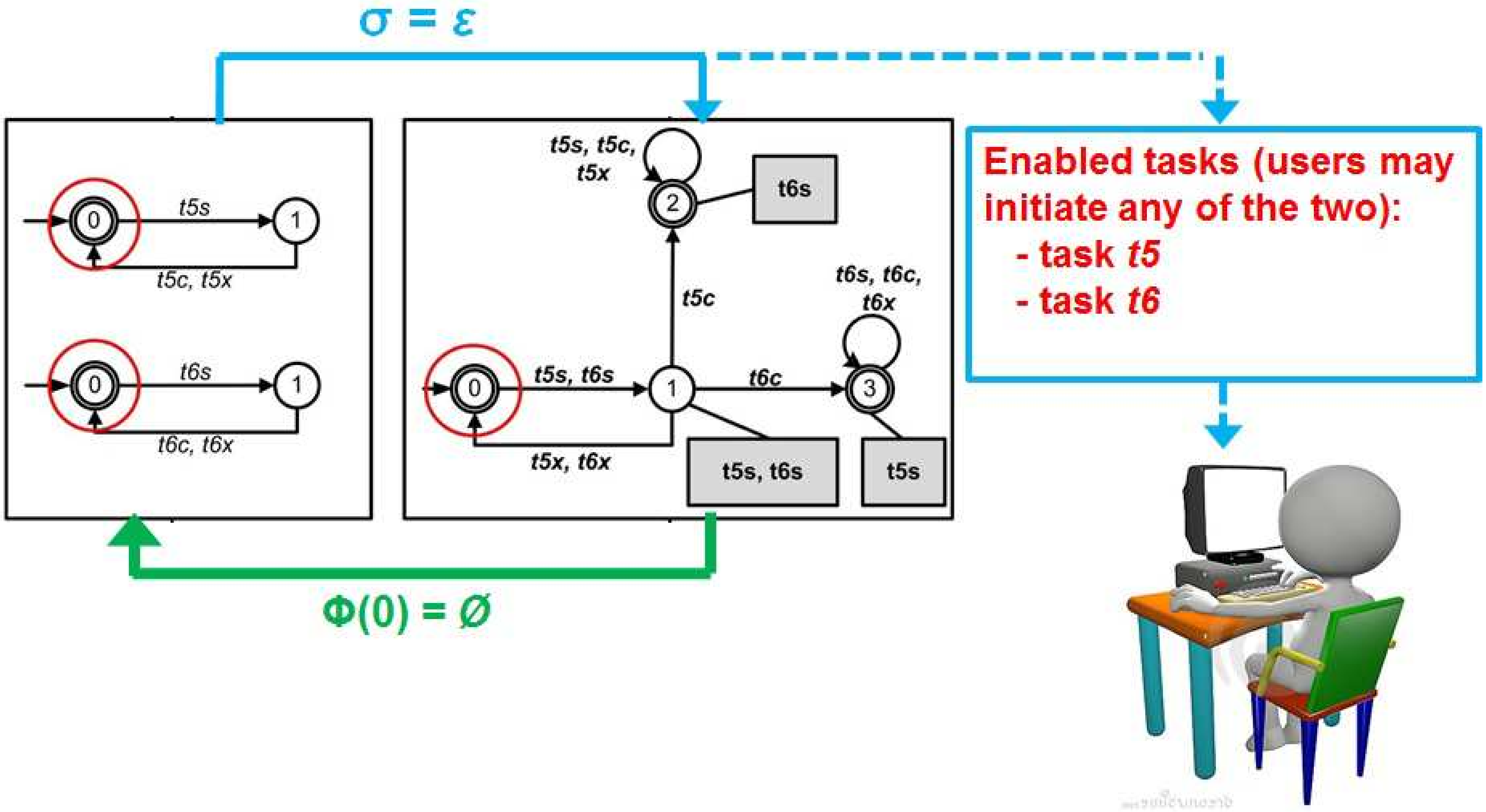
Supervisor (constraint 1of4)



Supervisor (constraint not coexistence)



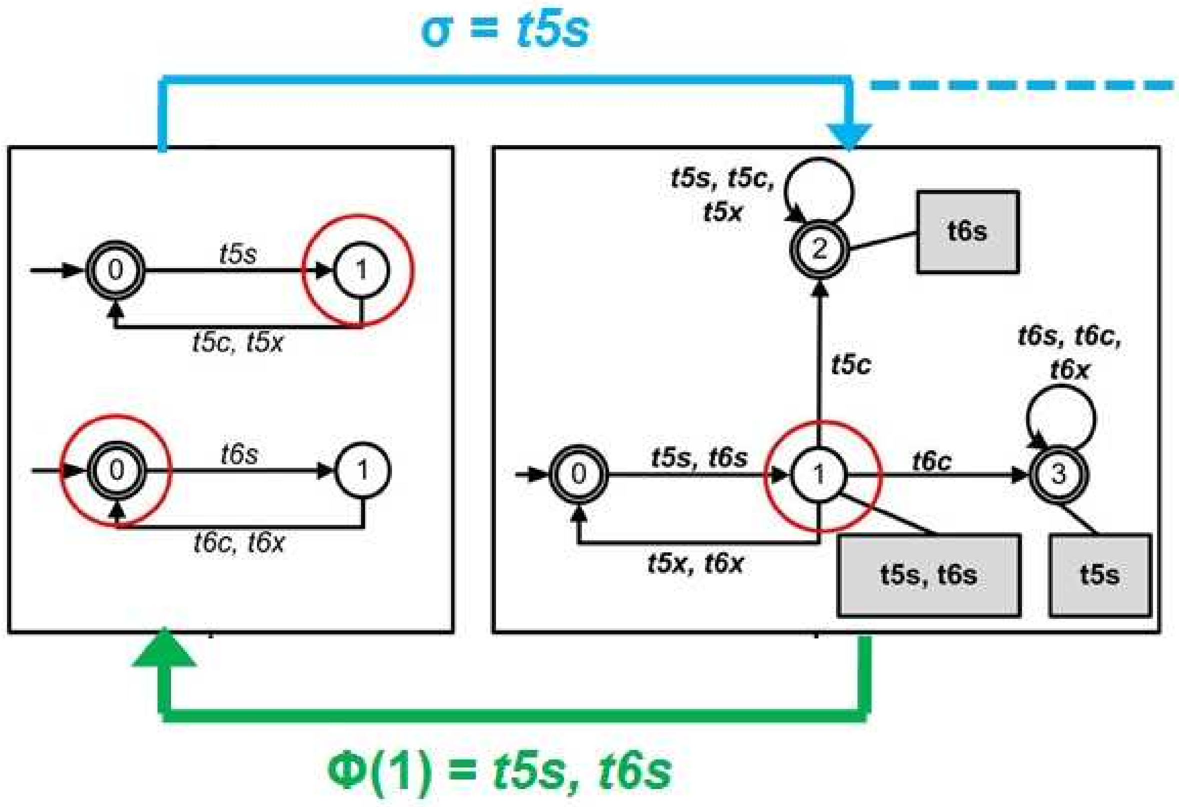
Example of application



Enabled tasks (users may initiate any of the two):

- task *t5*
- task *t6*

Example of application



After user has initiated task t5 (occurence of *t5s*), the only following actions are possible:

- complete *task t5*
- cancel *task t5*

Conclusions

What makes this approach good?

- Supervisory control theory allows a formal and automatic synthesis of supervisors. The obtained solution is correct by construction

- The supervisor restrains the process evolution, not allowing sequences which violates the constraints

- The obtained solution is minimally restrictive, in the sense that only is disabled or avoid what is really necessary

Conclusions

- In Constraint-based process it is difficult to envision all possible paths and the process are driven by user decisions rather than system decision

- It is difficult to model more abstract relations between tasks when the user has many choices in each state

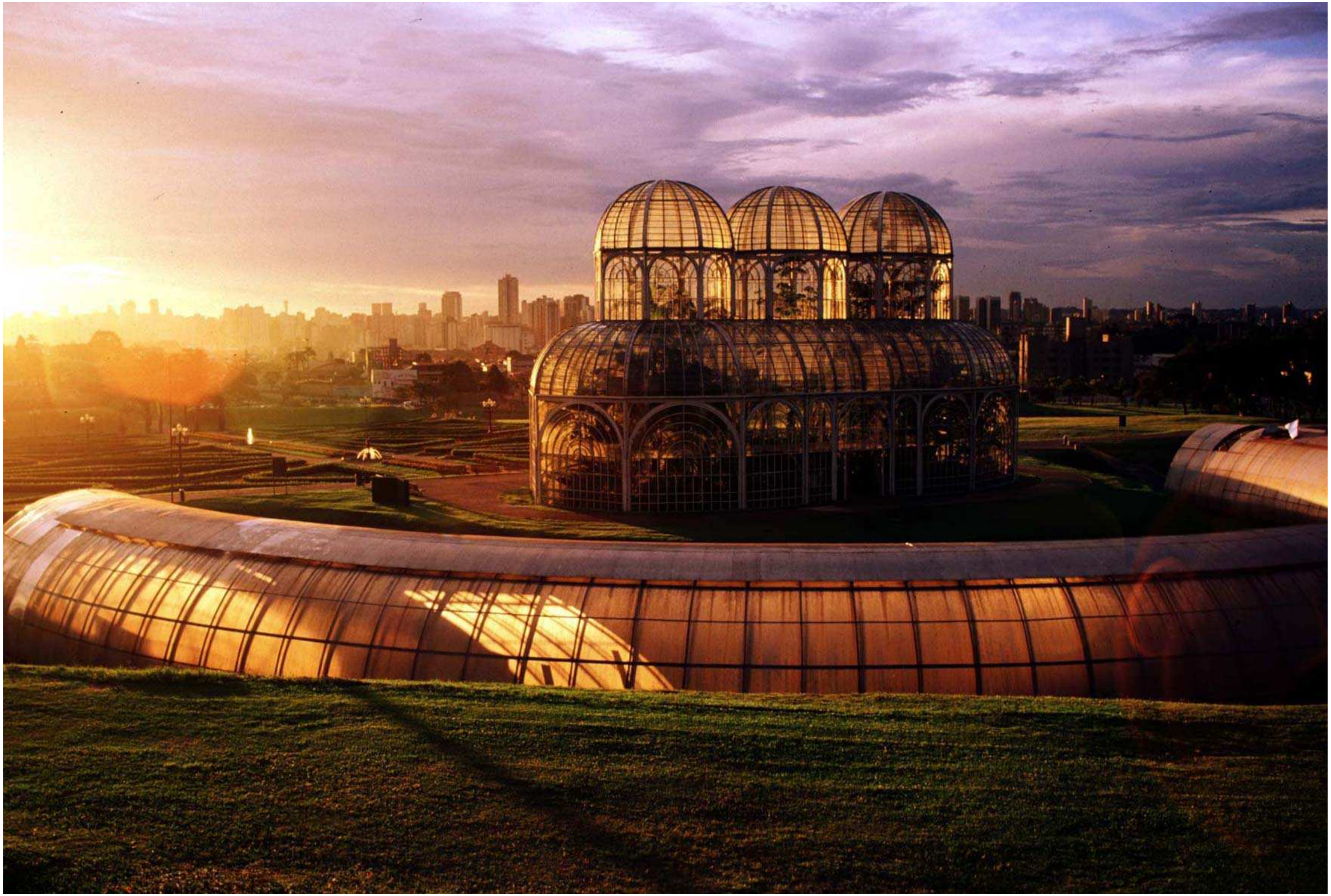
- Our approach aims to restrict execution sequences of tasks such that constraints are not violated. It does not limit the user by imposing rigid control-flow structures

Conclusions

- Ongoing works:
- Build a simulation environment of the supervisor architecture (considering a set of constraints constructs) (CPN Tools)
- Implementation of our approach in some information system

Thanks!

Curitiba, Brazil



PUCPR Campus, Curitiba, Brazil