Timed Automata Approach for the AXXOM Case Study (Scheduling a Lacquer Production)

VERIMAG

Plan

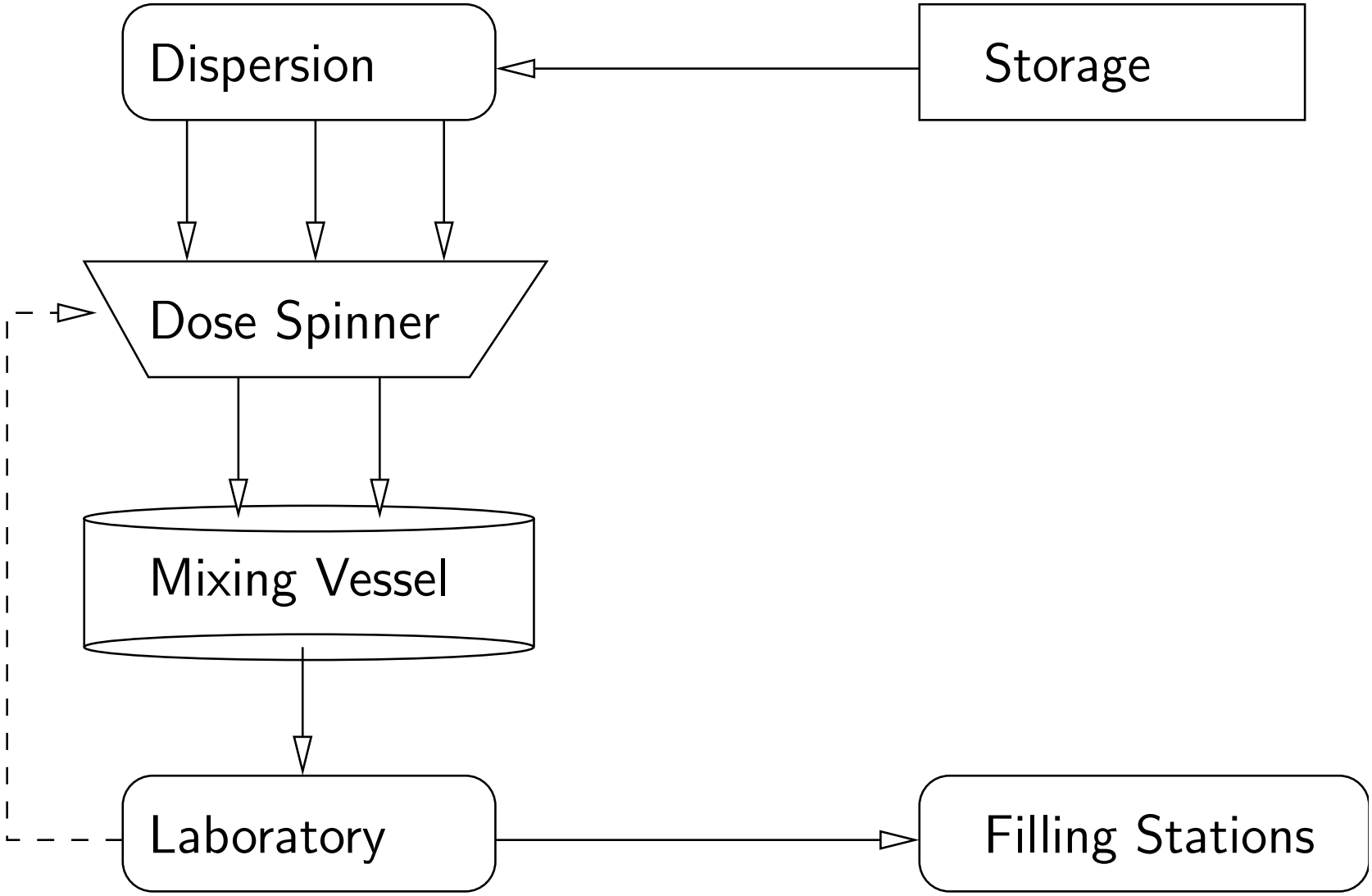
- problem characteristics
- modeling with timed automata
- finding an optimal schedule
- conclusions and future work

Problem characteristics (1/2)

- there are 29 lacquers to be produced, each one in some predefined time interval [earliest start date, due date]
- lacquers are of 3 different types, each type has a specific production flow characterized by the ressources involved, processing times, flow constraints, etc.
- lacquers have to be produced in different quantities and the processing times may depend on quantities; moreover quantities may exceed ressource capacities...

for each type of lacquer, we restrict ourselves to some fixed quantity which implies fixed processing times on ressources

Problem characteristics (2/2)



Modeling: resources

the number of available ressources for each ressource class (i.e., mixing vessel, dose spinner, etc.) is kept in a shared variable, initially:

tp	$= 1$	mix	$= 5$	abf	$= 2$
hdl	$= 1$	dvt	$= 1$	lab	$= 50$
dk	$= 2$	br	$= 1$		

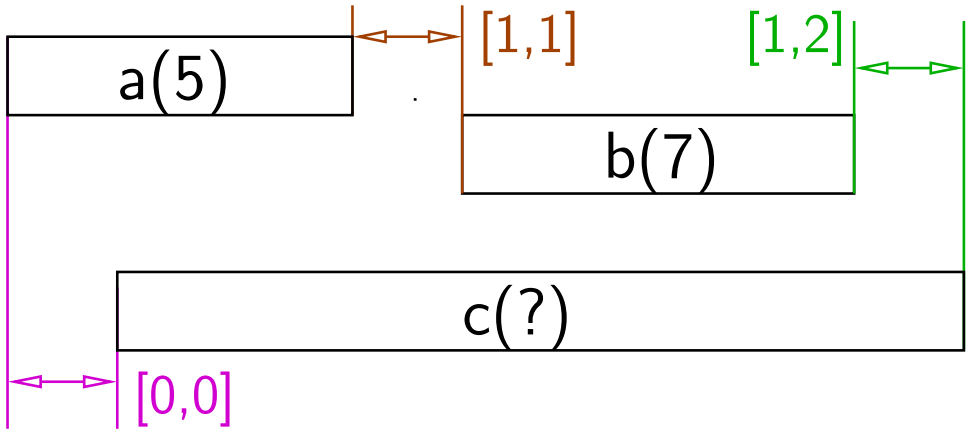
basic operations on ressources are straigtforward i.e., availability is $r > 0$, take is decrementation $r := r - 1$ and release is incrementation $r := r + 1$

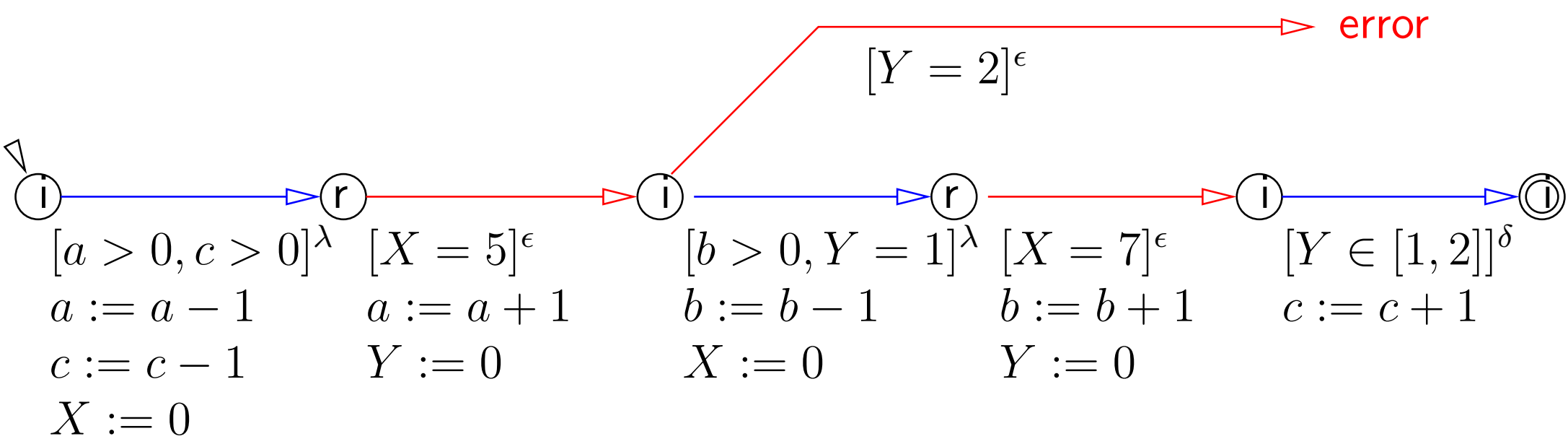
Modeling: lacquers (1/2)

each type of lacquer is modeled by a sequential timed automaton encoding :

- ressource allocation / deallocation order
- basic task duration constraints
- additional flow constraints (e.g, end-start, start-start, end-end)

Modeling: lacquers (2/2)





Modeling: production plan (1/2)

there are only three types of lacquers (uni, metallic and special metallic) but ones have to produce them for many times

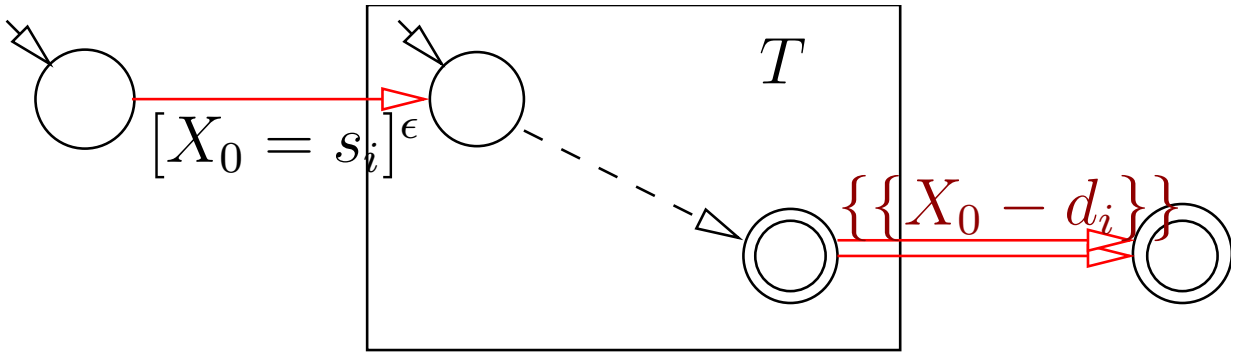
several copies of the timed automata corresponding are activated cf. the production plan, that is, a set of triples of the form

(lacquer type, earliest start date, due date)

Modeling: production plan (2/2)

let (t_i, s_i, d_i) be an item of the production plan and T the timed automaton corresponding to the lacquer type t_i.

our system contains the automaton T_i obtained from T as follows (X_0 is a global clock)



the final transition has the cost $\{\{X_0 - d_i\}\}$ denoting the extra-time needed by the job T_i to terminate

Modeling: wrap-up

The whole system is of the form $R \,||\, \prod_i T_i$

finding an optimal schedule is equivalent finding a minimum cost path leading to a (global) final state

but there are 29 different T_i's in the system which make the underlying state-space huge

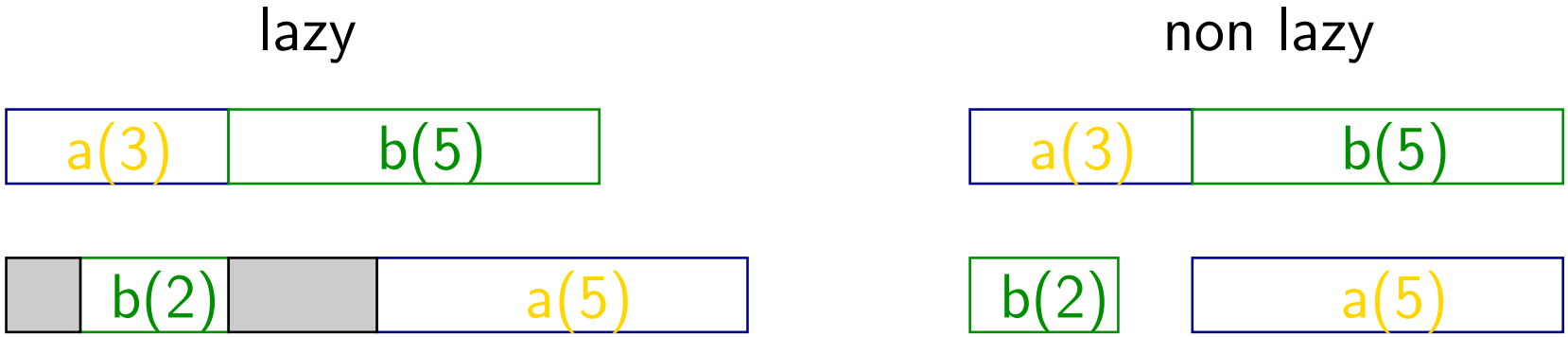
Heuristics

The idea is simple:

change the model such that the number of all paths get smaller but preserving (some of the) optimal ones, for instance

- explore only non-lazy runs i.e. remove useless waiting from executions
- forbid overtaking between lacquers of the same type i.e. enforce a first in, first out scheduling policy
- ensure minimal separation time between the starting times of lacquers of the same type

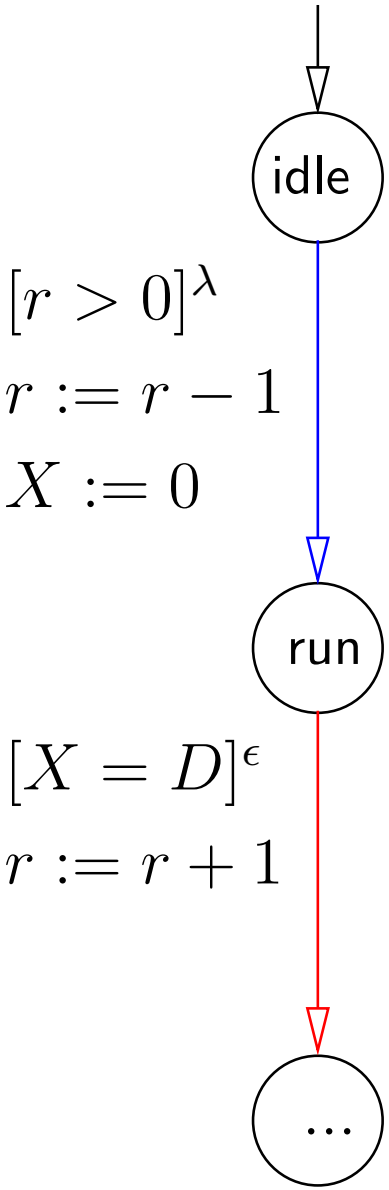
Heuristics - avoid lazy runs (1/2)

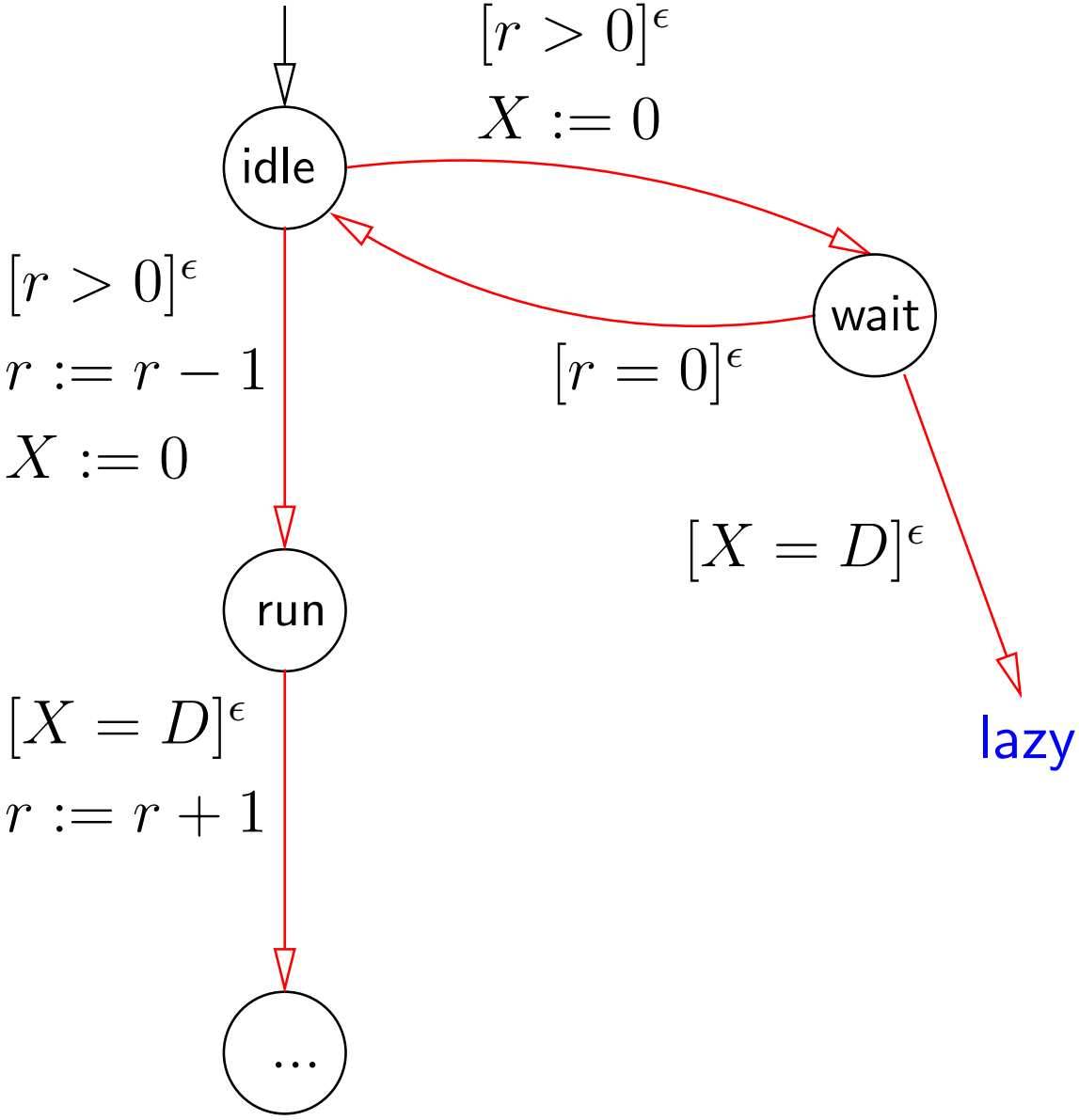


the general idea is exposed in [Abdeddaim-Maler-01]

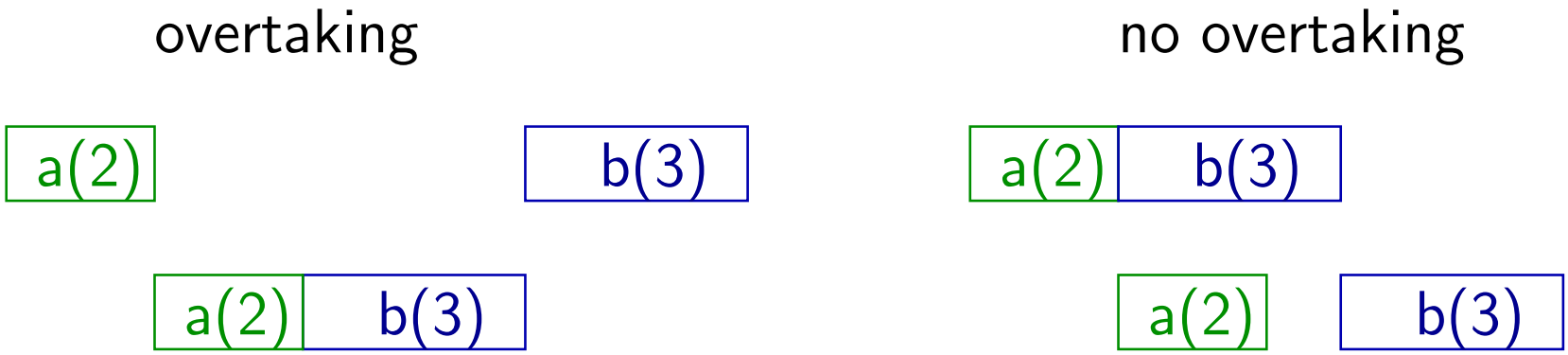
here, we consider a weaker(?) version which could be expressed syntactically in automata

Heuristics - avoid lazy runs (2/2)





Heuristics - no overtaking



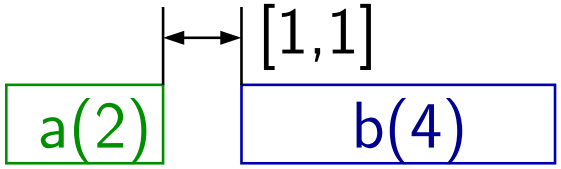
intuitively, all optimal paths are preserved...

we implement this restriction using asynchronous communication:

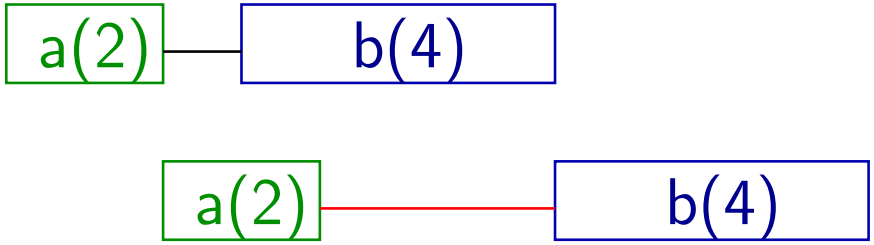
1. jobs of the same type are ordered according to the absolute start dates
2. each job signal its successor when it begins some phase
3. each job waits the (corresponding) signal from its predecessor before starting the same phase

n.b. this ensure a pipelined execution

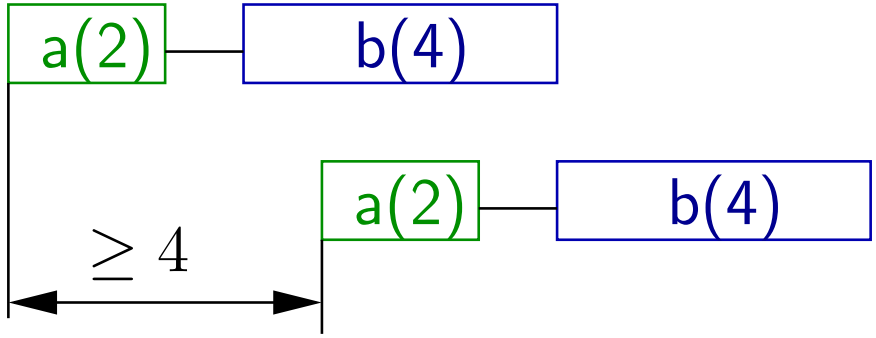
Heuristics - minimal separation time



non lazy $\longrightarrow$ constraint failure



enforce minimal separation



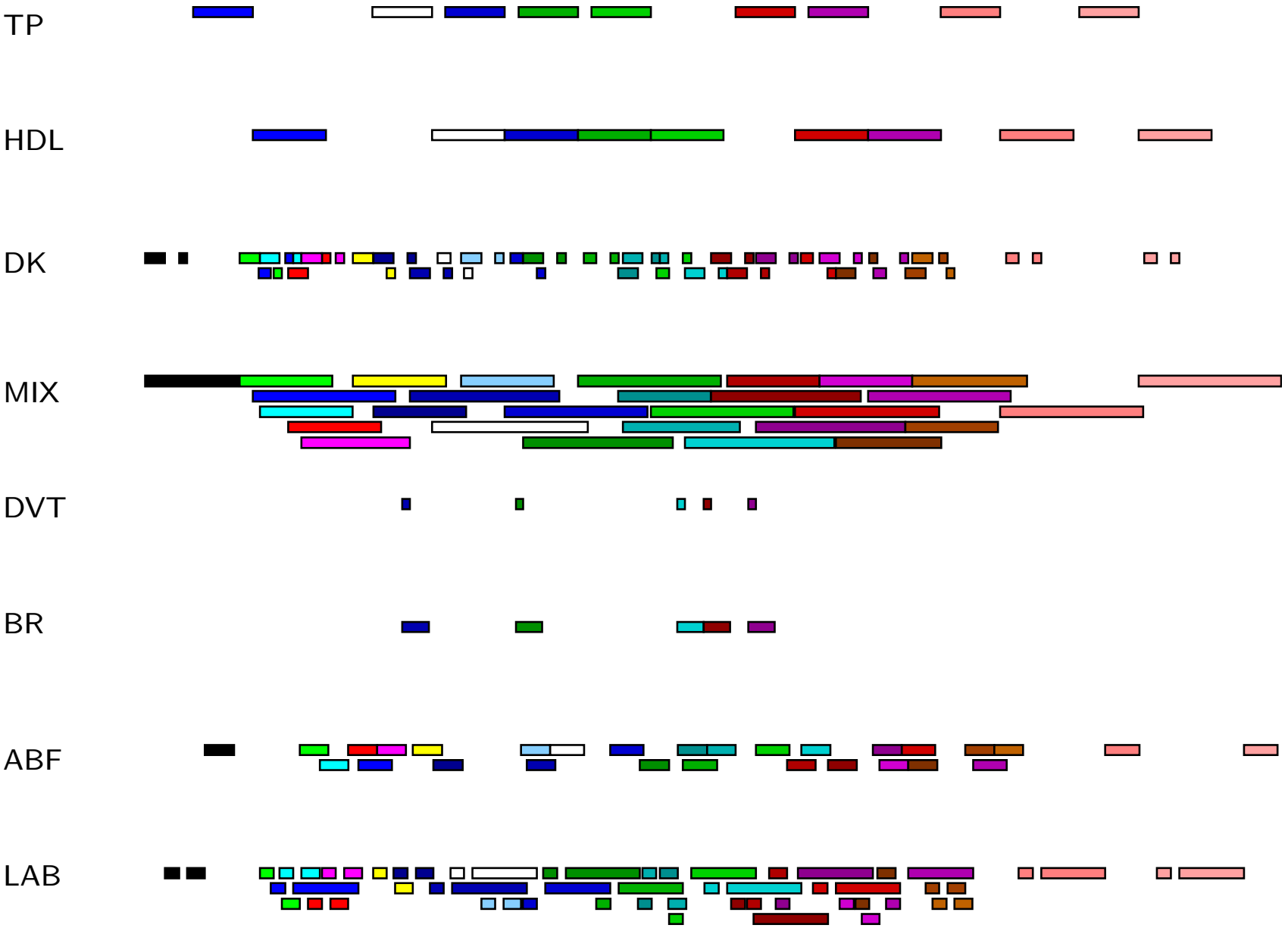
it is easy to extend the signaling mechanism which eliminate overtaking such that to ensure a safety distance: each job signal its successor a while after it begins some phase

n.b. in the example, if duration of a would be smaller than the duration of b there is no need for minimal separation time (i.e. it is equal to 0)

Finding an optimal schedule

- it takes $15''$ to find that a feasible schedule (having 0 delay) exists on the improved model
- such a schedule is extracted almost instantaeously by random execution (at depth 750)
- the state-space still cannot be constructed explicitly (more than $15 * 10^6$ states)

feasible schedule for 29 lacquers



Conclusions

- in general, problems could be solved using timed automata

Future work

- adding other quality measures
- batch splitting and/or merging