Petri nets and Linear logic as an aid for scheduling batch processes

Ronan Champagnat : L3i, Univ. La Rochelle
Brigitte Pradin-Chézalviel : LAAS-OLC Toulouse et IUT
Robert Valette : LAAS-OCSD Toulouse

http://www.laas.fr/~robert

Motivations (1)

• Batch processes : main features
  – Batches (discrete) of fluid (continuous)
  – Requires next resource before releasing preceding one => strong synchro. (deadlocks)
  – Operation durations are imprecise => need for robust solutions

• An approach currently implemented
  – Represent the recipe, resources, allocation policies by (deadlockfree) Petri nets
  – Heuristics for deriving starting dates and to solve conflicts
  – Simulation for evaluating the solution (are all the constraints satisfied)

• Some limitations of the approach
  – Difficulty of the interpretation of simulation results => how modifying decisions?
  – Is the schedule robust (consequences of operation duration variations)?
Motivations (2)

• Hybrid nature of scheduling
  – Logical: in which order are operations executed
  – Numerical: at which time point (dense time) are operations executed

• The two aspects are typically combined
  – Decide an order and derive dates
  – From dates derive the order

• Logic constraints are strong for batch processes
  – Require next resource before releasing the preceding one
  – Multi resource, storage require vessels etc.

→ Exploit logical constraints as much as possible

Our approach (1)

Principles

• Compute partial order between actions
  – Recipes, resource allocation, deadlock avoidance ⇒ represented by a Petri net
  – Build a proof of the corresponding linear logic sequent
    » derive partial order and algebraic expressions for dates and time interval
    » determine logical decisions which have to be made

• A kind of symbolic simulation
  – Timing considerations are stored as data attached to the tokens
  – Discrete event simulation, but no global time
  – It guarantees that causality among events is respected
  – It does not depend on specific numerical values (robust with respect to imprecision)
Our approach (2)

Linear logic (not linear temporal logic)

- Logical propositions are produced and consumed as tokens
- Based on sequent calculus (syntactical approach)
  - Hypotheses: initial marking+unordered list of transitions to be fired
  - Conclusions: final marking
  - We do not prove a formula, we prove a sequent (lin. logic describes the state changes)
  - It is the structure of the proof which provides the partial order

Our approach (3)

Linear logic and Petri nets: provability/reachability equivalence

\[ M \xrightarrow{t_1,t_2} M' \exists M'' M \xrightarrow{\bar{t}_1} M' \text{ et } M'' \xrightarrow{\bar{t}_2} M' \]

sequence: total order between transition firings, reachability

\[ M' = M + C.\top \]

unordered (number of time trans. fired), no reachability

\[ M,t_1,t_2 \vdash M' \]

unordered, it is derived, if proved then: reachability
Our approach (4)

- Fragment MILL (Multiplicative, Intuitionist LL)
- Conjunction of resources $A \odot A \& C, A \odot B$
  - Markings, Preconditions, Postconditions
- Linear Implication $A \odot B \rightarrow C \& D$
  - Transition, consuming/production, causality
- Sequent: firing $A \odot B, A \odot B \rightarrow C \& D \leftarrow C \& D$

Our approach (5)

What is the benefit of a logic based approach?

3 basic rules for linear logic sequent calculus for Petri nets:

- cut rule, derive (fire) in sequence $\Rightarrow$ token player
- right introduction of $\otimes$, fire in parallel
- left introduction of $\rightarrow$, causality
Scheduling (1)

• Proof tree based on causality (left intro. linear implic.)
• Time stamps are attached to the tokens
• Formal computation of time stamps (max, +)
• Conflicts - decision making - deriving a partial order
  – Token conflict
    » more than one token can be used to fire a transition
  – Transition conflict
    » at least two transitions share a token

Scheduling (2)

• Duration of a scenario
  – Between initial and final marking
  \[ A \quad t_1 \quad B \quad t_2 \quad C \]

• Date of an event

• Time interval during which:
  – a token (resource) is available (a state is true)
  – a set of tokens (resources) (state)
    are concurrently present in some places

• Duration are variables (necessary/possible)
Example (1)

Petri net modeling

- States of parts and resources
- Events: Decisions (start op.)
- Events: End of operations (duration)
- Markings (initial and final)
  - Defines one scheduling problem

Example (2)

- Conflicts "t11 and t23", "t21 and t13"
- Dt11 = 0 and Dt21 = 0
  - C1 produced at: dt14 + max(dt12,dt22)
  - C2 produced at: dt24 + max(dt12,dt22)
- Dt11 = 0 and Dt13 = dt12
  - C1 produced at: dt12 + dt14
  - C2 produced at: dt12 + dt14 + dt22 + dt24
- Dt21 = 0 and Dt23 = dt22
  - C1 produced at: dt12 + dt14 + dt22 + dt24
  - C2 produced at: dt22 + dt24
Conclusion

• Generate sets of continuous constraints
  – Consistent with the discrete ones (one partial order)
  – Two objectives:
    » to make one decision (preceding ones in causal order are taken)
    » to analyze the influence of operation durations for ONE partial order

• Continuous constraints are derived by formal computations
  – Operation durations may be explicit functions of token attributes (batch size)

• Efficient when discrete constraints are strong
  – A lot of zero delay constraints and large imprecision about operation durations

Our approach ()

• In sequence
  \[ A \otimes B, t_1 \vdash B \otimes B \leadsto A \otimes B, t_1, t_2 \vdash B \otimes C \]

• In parallel
  \[ A, t_1 \vdash B \otimes B, t_2 \vdash C \leadsto A \otimes B, t_1, t_2 \vdash B \otimes C \]

• Causality
  \[ A \vdash A \otimes B, B \otimes B, t_1 \vdash B \otimes C \quad \text{decision} \]
  \[ A, B, t_1, t_2 \vdash B \otimes C \quad \text{ol.} \]
Scheduling ()

Conflict between tokens
– choosing one token or the other one in C results in an other duration
– a scheduling decision is required

\[ \text{scenario } s : A \otimes B, t_1, t_2, t_3 \vdash C \otimes D \]
\[ \text{duration } \left( \max(d_1, d_2 + d_3) \right) \text{ or } \left( \max(d_2, d_1 + d_3) \right) \]
\[ (t_1 \mid (t_2 ; t_3)) \text{ or } (t_2 \mid (t_1 ; t_3)) \]

(t_2 \text{ before } t_3) \text{ or } (t_1 \text{ before } t_3) \text{ to define a scenario and a duration}

Scheduling ()

Conflict between transitions
– fire first \( t_1 \) or \( t_2 \), a scheduling decision

\[ \text{scenario } s : A \otimes B \otimes R, t_1, t_2, t_3, t_4 \vdash E \otimes F \otimes R \]
\[ \text{duration } \left( d_1 + \max(d_3, d_2 + d_4) \right) \text{ or } \left( d_2 + \max(d_4, d_1 + d_3) \right) \]
\[ (t_1 : (t_3 \mid (t_2 ; t_4))) \text{ or } (t_2 : (t_4 \mid (t_1 ; t_3))) \]

It is necessary to add a new partial order constraint in order to completely define a scenario