A Logical Framework for Workflow Scheduling under Resource Allocation Constraints

Pinar Senkul
(Middle East Technical University, Ankara, Turkey)

Michael Kifer
(Stony Brook University, USA)

Ismail H. Toroslu
(Middle East Technical University, Ankara, Turkey)
Problem

Workflow Scheduling:

Find execution sequence of tasks in a workflow so that all constraints are satisfied
Definitions

**Workflow:**
- A coordinated set of activities that act together to achieve a well-defined goal

**Constraints:**
- *Temporal/causality*
  - Do t1 only after t2 and t3 have been executed
  - Do t1 if t2 has been or will be executed
- *Resource allocation*
  - Execute workflow keeping the cost below threshold
  - Don’t use same machine for concurrent subtasks in workflow
Problem (contd.)

- Most works deal with temporal/causality constraints
- We deal with resource allocation constraints
Example: House Construction Workflow

- The budget should not exceed the given amount
- The construction should not last longer than the given duration
- Different companies must be chosen for parallel tasks
Approaches to Workflow Scheduling

- Operations Research
- AI
- Petri Nets
- Logic (temporal, concurrent transaction logic)
Our Approach

CCTR – *Concurrent Constraint Transaction Logic*

- Integrates Concurrent Transaction Logic (CTR) with Constraint Logic Programming (CLP)
  - Gives precise semantics to workflows
- Can
  - Model
  - Reason about
  - Schedule
workflows
Overview of Concurrent Transaction Logic

- Extension of first order logic for programming, executing and reasoning with state changing concurrent processes.

- Syntax

  \[ \text{wall} \otimes ( ( \text{carpentry} \otimes \text{roof} ) \mid \text{installation} ) \]

  execute wall then in parallel do:
  - carpentry followed by roofing installation work

- Semantics – paths

  \[ \langle D_0 D_1 D_2 D_3 D_4 \rangle \]

  any interleaving of \text{carpentry}\otimes\text{roof} and \text{installation} is a valid execution
Concurrent *Constraint Transaction Logic* (CCTR)

- **syntax**: same as in CTR

- **semantics**: 
  *partial schedule* is more structured than a path, parallel and serial parts of the execution are explicitly shown.

  e.g. \( \langle D_0 D_1 \rangle \cdot_p \langle D_1 D_2 \cdot_p D_2 D_3 \rangle \parallel_p \langle D_1 D_4 \rangle \)
Resources

*resource*: an object with the attributes *cost, token*

*resource assignment*: a partial mapping

*partial schedules* $\rightarrow$ *set of resources*
A partial schedule satisfies a resource allocation constraint $c$ if

$$c(\text{partial schedule, resource assignment})$$

is true.

A constraint system is a set of constraint definitions. It consists of

- **cost subsystem**: contains definitions of cost constraints
- **control subsystem**: contains definitions of control constraints
Cost Subsystem - House Construction Workflow

Template: generic procedure for computing costs over schedules

\[
\begin{align*}
\text{cost\_constraint}(\omega, \text{asg}) &= \text{value\_constraint}(\text{cost}(\omega, \text{asg})) \\
\text{cost}(\omega_1 \cdot p \omega_2, \text{asg}) &= \text{op}_\otimes(\text{cost}(\omega_1, \text{asg}), \text{cost}(\omega_2, \text{asg})) \\
\text{cost}(\omega_1 \parallel p \omega_2, \text{asg}) &= \text{op}_\vert(\text{cost}(\omega_1, \text{asg}), \text{cost}(\omega_2, \text{asg}))
\end{align*}
\]

Problem-specific part for the constraint:
"construction should not exceed the given duration"

\[
\begin{align*}
\text{cost}(\omega, \text{asg}) &= \text{cost\_of}(\text{asg}(\omega)) \\
\text{value\_constraint}(V) &= V < c \\
\text{op}_\vert(V_1, V_2) &= \max(V_1, V_2) \\
\text{op}_\otimes(V_1, V_2) &= V_1 + V_2
\end{align*}
\]
Control Subsystem - House Construction Workflow

**Template:**

*generic procedure for computing control constraints*

\[ \text{ctrl}\_\text{constraint}(\omega_1 \cdot p \omega_2, \text{asg}) = \]

\[ \text{set}\_\text{constraint}(\otimes (\text{asg}(\omega_1), \text{asg}(\omega_2)), \]

\[ \text{ctrl}\_\text{constraint}(\omega_1, \text{asg}), \text{ctrl}\_\text{constraint}(\omega_2, \text{asg})] \]

\[ \text{ctrl}\_\text{constraint}(\omega_1 \parallel p \omega_2, \text{asg}) = \]

\[ \text{set}\_\text{constraint}(\sqcap (\text{asg}(\omega_1), \text{asg}(\omega_2)), \]

\[ \text{ctrl}\_\text{constraint}(\omega_1, \text{asg}), \text{ctrl}\_\text{constraint}(\omega_2, \text{asg})] \]

\[ \text{ctrl}\_\text{constraint}(\omega, \text{asg}) = \text{leaf}\_\text{constraint}(\text{asg}(\omega)) \]

**Problem specific part for the constraint:**

"Different subcontractors must be chosen for parallel tasks"

\[ \text{set}\_\text{constraint}(R_1, R_2) = (\text{token}\_\text{of}(R_1) \cap \text{token}\_\text{of}(R_2) = \emptyset) \]
Logical Entailment in Constraint System

\[ M, \zeta, \omega, \text{asg} \models \varphi \]

iff

\[ M, \omega \models \varphi, \]
\[ \mathcal{D} \models \text{cost\_constraint}(\omega, \text{asg}) \text{ and } \]
\[ \mathcal{D} \models \text{ctrl\_constraint}(\omega, \text{asg}) \]
Using CCTR as a Workflow Scheduler

- Take a CCTR workflow which includes constraints
- Construct an equivalent CTR workflow so that:
  - There is a 1-1 correspondence between the partial schedules of the CCTR workflow and the resulting CTR workflow
  - These partial schedules & resource assignments for them are computed by the proof theory of CTR
  - Can now schedule conjunction-free workflows using CTR interpreter
  - Can take advantage of prior work on scheduling workflows under causality/temporal constraints
Transformation Rules

- $B(G_1 \lor G_2) = B(G_1) \lor B(G_2)$
- $B(G) = R(G,T) \otimes \text{cost\_constraint}(T) \otimes \text{ctrl\_constraint}(T)$
- $R(A,T) = A \otimes (T=\text{resource\_asg}(A,\text{Agents}))$
- $R(G_1 \mid G_2, T) = (T='\mid'(T_1,T_2)) \otimes (R(G_1,T_1) \mid R(G_1,T_2))$
- $R(G_1 \otimes G_2, T) = (T='\otimes'(T_1,T_2)) \otimes (R(G_1,T_1) \mid R(G_1,T_2))$
- $R(G_1 \lor G_2, T) = R(G_1,T) \lor R(G_2,T)$

- basically structural induction with insertion of constraints in appropriate places
Transformation for Part of House Construction Example

\[
((\text{carpentry} \otimes \text{roof}) \mid (\text{installation} \mid \text{ceiling})) \land \\
\text{cost\_constraint} \land \text{ctrl\_constraint}
\]

\[
T = |'(T_1, T_2) \otimes \\
(T_1 = '(\otimes' (T_3, T_4)) \otimes \\
((\text{c} \otimes T_3 = \text{rsr(c,W)}) \otimes \\
(\text{r} \otimes T_4 = \text{rsr(r,X)}))))) \\
| \\
(T_2 = |'(T_5, T_6) \otimes \\
((\text{i} \otimes T_5 = \text{rsr(i,Y)}) \mid \\
(\text{e} \otimes T_6 = \text{rsr(e,Z)}))))) \\
\otimes \text{cost\_constraint}(T) \otimes \text{ctrl\_constraint}(T)
\]
Constraint Template – Cost Constraints

-Logic rules that mimic the definition of the cost part of a constraint system

**Template:**

```prolog
cost_constraint(T) :- cost(T,V), value_constraint(V)
cost((T_1, T_2), V) :- cost(T_1, V_1), cost(T_2, V_2), op_(V_1, V_2, V)
cost(⊗(T_1, T_2), V) :- cost(T_1, V_1), cost(T_2, V_2), op_⊗(V_1, V_2, V)
cost(resource_asg(T, Agents), V) :- cost_of(resource_asg(T, Agents), V)
```

**User-defined terms and predicates:**

```prolog
resource_asg(T, A) := rsrc(T, A)
cost_of(rsrc(T, A), V) :- duration(T, A, V)
value_constraint(V) :- V < c
op_(V_1, V_2, V) :- V = max(V_1, V_2)
op_⊗(V_1, V_2, V) :- V = V_1 + V_2
```

Ensures that duration is within constraints
Constraint Template-Control Constraints

-Same for the control part of a constraint system

**Template:**

\[
\text{ctrl\_constraint}(\| (T_1, T_2)) \ :- \ \text{set\_constraint}(T_1, T_2),
\]  
\[
\text{ctrl\_constraint}(T_1), \ \text{ctrl\_constraint}(T_2)
\]

\[
\text{ctrl\_constraint}(\otimes (T_1, T_2)) \ :- \ \text{set\_constraint}(\otimes (T_1, T_2),
\]  
\[
\text{ctrl\_constraint}(T_1), \ \text{ctrl\_constraint}(T_2)
\]

\[
\text{ctrl\_constraint}(T) \ :- \ \text{leaf}(T), \ \text{leaf\_constraint}(T)
\]

**User-defined predicates:**

\[
\text{set\_constraint}(\| (T_1, T_2)) \ :- \ \text{disjoint}(T_1, T_2)
\]

\[
\text{set\_constraint}(\otimes (T_1, T_2)) \ :- \ \text{true}
\]

\[
\text{leaf\_constraint}(T) \ :- \ \text{true}
\]

*Ensures that resources are not shared on parallel paths*
Correctness of the Transformation

**Theorem:**

Let $\zeta$ be a constraint system and $\mathcal{D}$ be a constraint domain, which can be represented using the template rules. Then, the transformation $B$ is a correct prescheduler (i.e., $B$ yields equivalent workflows that satisfy constraints)
The Big Picture

- Rsnc asg term
- G Workflow
- G' = B(G)
- CTR Interpreter
- Const. def.
- Const. set
- Schedule
- Constraint Solver
- Resource assignment
Conclusion

- Presented a logical framework for **modelling** and **scheduling** workflows under resource allocation constraints (implemented using a CTR interpreter and XSB)

- This framework can be extended with special-purpose constraint solvers optimised for our framework

- Scheduling under resource allocation constraints for dynamically changing workflows and for multiple concurrent instances of the same workflow are topics for future work