Introduction to the Process Specification Language

Conrad Bock
U.S. National Institute of Standards and Technology

April 7, 2009
“PSL” stands for …

- … Process Specification Language,
- … but not in the sense of a process model or programming language.

Should be called:
- Process Semantics Language.
- Process Constraint Language.

PSL enables:
- Describing what actually happens when a process specification executes.
- Writing constraints on processes.
PSL Background

- Based on long period of research:
  - Situation calculus.
  - Process Interchange Format (PIF).
  - Enterprise modeling.

- Applied to scheduling, process modeling, process planning, production planning, simulation, project management, workflow, business process reengineering, vehicle navigation, semantic interoperability.

- ISO 18629, full international standard.

- Basis of Semantic Web Services (SWSL) at W3C.


Process Models

- **UML 2:**

  ```
  ChangeColor
  ```

  ```
  ![Diagram](image)
  ```

- **BPEL:**

  ```
  <process name="ChangeColor">
    <sequence>
      <invoke operation="Paint"/>
      <invoke operation="Dry"/>
    </sequence>
  </process>
  ```

- **C:**

  ```
  void ChangeColor
  { Paint();
    Dry();
  }
  ```
What Happens?

- Does Dry start only after Paint finishes?
  - Yes, unless Paint is invoked asynchronously.

- Does Dry happen everytime Paint does?
  - Not necessarily, the model is referring only to the ChangeColor process.

- These questions are about the actual execution of the process.
  - Which steps start and stop when.
  - What the process model “means”.
Capture the Meaning

- How do we know what the model means?
  - Read the language documentation.
  - Execute the model on a reference implementation.
  - Ask somebody.

- Humans eventually figure it out, but what about the tools?
  - They can’t read documentation, experiment with implementations, or ask anyone.

- Need a way to express the meaning of process models in a way tools can understand.
Model vs. Execution

- Each satisfying the constraints of the model.

What the model “means”
Occurrence is an execution of an Activity
– for example, Paint executed at 10:22am ET 9/1/2003 at factory 1.
PSL Model of Execution

- PSL is defined in the Common Logic Interchange Format (CLIF)...
- ... but could be OCL or other FOL language.

\[
\begin{align*}
&\text{(forall } (?occ \ ?a) \\
&\text{  (if } (\text{occurrence_of } ?occ \ ?a) \\
&\text{    (and } (\text{activity_occurrence } ?occ) \\
&\text{      (activity } ?a)))))
\end{align*}
\]

\[
\begin{align*}
&\text{(forall } (?occ) \\
&\text{  (if } (\text{activity_occurrence } ?occ) \\
&\text{    (exists } (?a) \\
&\text{      (and } (\text{activity } ?a) \\
&\text{        (occurrence_of } ?occ \ ?a) )))
\end{align*}
\]

\[
\begin{align*}
&\text{(forall } (?occ \ ?a1 \ ?a2) \\
&\text{  (if } (\text{and } (\text{occurrence_of } ?occ \ ?a1) \\
&\text{       (occurrence_of } ?occ \ ?a2) \\
&\text{      (= } ?a1 \ ?a2)))))
\end{align*}
\]

Executions happen one after another.

Notice the multiplicities:

- Occurrence has multiple successors, one for each (theoretically) possible next occurrence.
Anything Can Happen

- Tree of all possible execution sequences, including
  - not physically possible.
  - not specified by the user.
- Not stored anywhere, just referred to by constraints.
Rules as Occurrence Constraints

- Specify which parts of the occurrence tree are “legal”.
- Example rule: drying immediately follows all painting.
**Constraint Language**

- **In CLIF:**

  \[
  \text{forall} \ (?\text{occPaint}) \\
  \text{if} \\
  \text{and} \ (?\text{occPaint} \text{ Paint}) \\
  \text{legal} \ (?\text{occPaint}) \\
  \exists \ (?\text{occDry}) \\
  \text{and} \ (?\text{occDry} \text{ Dry}) \\
  \text{legal} \ (?\text{occDry}) \\
  \text{forall} \ (?\text{otherSuccessor}) \\
  \text{if} \\
  \text{not} \ (= \ ?\text{otherSuccessor}) \\
  \text{successor} \ (?\text{occPaint}) \\
  \text{legal} \ (?\text{otherSuccessor})
  \]

**Legend:**
- **red** = first order logic
- **occurrence_of** = PSL
- **black** = engineer’s process
Processes in PSL

- Above says that Dry happens after Paint under executions of ChangeColor.
- Outside of ChangeColor Paint can occur without Dry.
Processes as Occurrence Constraints

- Paint happens immediately after Dry under executions of ChangeColor.
- ChangeColor specification does not constrain OtherProcess above.
Complex Processes in PSL

- Complex occurrences and activities composed of primitive ones:

  - Successor moved to PrimitiveOccurrence.
  - Occurrence tree covers every step at finest level of granularity.
Complex Processes in PSL

- Execution sequencing within complex activity:

  - (Defined in terms of successor)

  - Occurrences following *immediately*
    - min_precedes
  - Occurrences following *sometime, not necessarily immediately.*
Complex Processes in PSL

- Constrain occurrences of ChangeColor to be composed of sequential occurrences of Paint and Dry:

\[
\text{(forall } (?\text{occChangeColor})
\text{ (if}
\text{ (occurrence_of } (?\text{occChangeColor} \text{ ChangeColor})
\text{ (exists } (?\text{occPaint } ?\text{occDry})
\text{ (and (occurrence_of } (?\text{occPaint} \text{ Paint})
\text{ (occurrence_of } (?\text{occDry} \text{ Dry})
\text{ (subactivity_occurrence } (?\text{occPaint} ?\text{occChangeColor})
\text{ (subactivity_occurrence } (?\text{occDry} ?\text{occChangeColor})
\text{ (next_subocc } (?\text{occPaint } ?\text{occDry}
\text{ ChangeColor)))))}
\]
Rule / Process Consistency

- Rule: drying immediately follows all painting.
- Process:

ChangeColor2 occurrences do not satisfy rule.

Where rule is satisfied, ChangeColor2 is not.
Behavior Classification

How to abstract commonality?

Behavior Classification

- **Food Service** has these steps:
  - Order, Prepare, Serve, Eat, Pay

- **With these constraints:**
  - Order, Prepare, and Serve always happen before Eat.
  - Serve happens after Prepare and Order.
  - Pay can happen anytime in the process.

- **Fast Food Service** adds:
  - Prepare before Order.

- **Need to partially specify a process ...**
- ... and incrementally build up constraints.
Behavior Classification

- Flow models are not expressive enough:

- Prepare and Order are not concurrent.
- Pay is not concurrent with other steps.
Behavior Classification

- **FoodService**: Prepare sometime before Eat.

\[
(\text{forall } (?\text{occFoodService}) \\
(\text{if} \\
(\text{occurrence\_of } ?\text{occFoodService} \text{ FoodService}) \\
(\text{exists}(?\text{occPrepare} ?\text{occEat}) \\
(\text{and} \\
(\text{occurrence\_of } ?\text{occPrepare} \text{ Prepare}) \\
(\text{occurrence\_of } ?\text{occEat} \text{ Eat}) \\
(\text{subactivity\_occurrence } ?\text{occPrepare} ?\text{occFoodService}) \\
(\text{subactivity\_occurrence } ?\text{occServe} ?\text{occFoodService}) \\
(\text{min\_precedes } ?\text{occPrepare} ?\text{occEat} \text{ FoodService}))))
\]
**Behavior Classification**

### FastFoodService: Prepare sometime before Order.

```prolog
(forall (?occFastFoodService)
    (if
        (occurrence_of ?occFastFoodService FastFoodService)
        (exists (?occPrepare ?occOrder ?occFoodService)
            (and
                (occurrence_of ?occPrepare Prepare)
                (occurrence_of ?occOrder Order)
                (subactivity_occurrence ?occPrepare ?occFoodService)
                (subactivity_occurrence ?occOrder ?occFoodService)
                (min_precedes ?occPrepare ?occOrder FoodService))))
```
Behavior Classification

Classification of process executions:
- All subactivity occurrences of FastFoodService occurrences are subactivity occurrences of FoodService occurrences.

\[
\text{(forall} \ (?\text{occFFS}) \\
\quad \text{(if} \ (?\text{occFFS} \text{FastFoodService}) \\
\quad \quad \text{(exists} \ (?\text{occFS}) \\
\quad \quad \quad \text{(and} \ (?\text{occFS} \text{FoodService}) \\
\quad \quad \quad \quad \text{(forall} \ (?s) \\
\quad \quad \quad \quad \quad \text{(if} \ (?\text{occFFS} ?s) \\
\quad \quad \quad \quad \quad \quad \text{(subactivity_occurrence} \ ?s ?\text{occFFS}))))))
\]
Behavior Classification

Execution traces conforming to general and/or specific process constraints, or not.
Behavior Classification

- Possible enhancement to UML notation.
  - (Requires updating tools and services ...)
  - ... compared to extending CLIF representation)
Web Service Queries

- **Buy a book:**
  - without using a credit card.
  - credit card charged only when shipped.

(adapted from example by Michael Gruninger)

- **Shipping:**
  - transport frozen vegetables from San Francisco to DC.

- **Substituting:**
  - a web service with another that achieves the desired effects at lower cost.
Web Service Queries

- Web service posts specification of the public aspects of their process.
- Query is a specification of the desired aspects of a process.
- Answer tells which web service are consistent with the query.
  - Which posted specifications are generalizations of the query?
Web Service Queries

- Buy a book without using a credit card.
  
  $$\text{(forall } (?\text{occ})$$
  $$\quad (\text{if } (\text{occurrence\_of } ?\text{occ }\text{DesiredProcess})$$
  $$\quad \quad (\text{and } (\text{exists } (?s1)$$
  $$\quad \quad \quad (\text{and } (\text{occurrence\_of } ?s1 \text{ ShipBook})$$
  $$\quad \quad \quad \quad (\text{subactivity\_occurrence } ?s1 \text{ ?occ})))$$
  $$\quad \quad \quad (\text{not } (\text{exists } (?s2)$$
  $$\quad \quad \quad \quad (\text{and } (\text{occurrence\_of } ?s2 \text{ ChargeCreditCard})$$
  $$\quad \quad \quad \quad \quad (\text{subactivity\_occurrence } ?s2 \text{ ?occ}))))))$$

- ... with credit card charged after ship.
  
  $$\text{(forall } (?\text{occ})$$
  $$\quad (\text{if } (\text{occurrence\_of } ?\text{occ }\text{DesiredProcess})$$
  $$\quad \quad (\text{exists } (?s1 ?s2)$$
  $$\quad \quad \quad (\text{and } (\text{occurrence\_of } ?s1 \text{ ShipBook})$$
  $$\quad \quad \quad \quad (\text{subactivity\_occurrence } ?s1 \text{ ?occ})$$
  $$\quad \quad \quad \quad (\text{occurrence\_of } ?s2 \text{ ChargeCreditCard})$$
  $$\quad \quad \quad \quad (\text{subactivity\_occurrence } ?s2 \text{ ?occ})$$
  $$\quad \quad \quad \quad (\text{min\_precedes } ?s1 \text{ ?s2 }\text{DesiredProcess}))))$$
Refinement Rollback

- Alternative process commitments.
Protocol Processes

- **Constraints on messages (as subprocesses)**
  - After first message arrives at Org 2, second two are sent in parallel to Org 1.
  - After those both arrive at Org 1, last message is sent to Org2.

- **For defining standard or contractual interactions (eg, RosettaNet).**
Process / Rule Consistency

- Business rules and processes are usually represented in incomparable languages.
- In PSL, they are both constraints on processes.
- Can automatically check consistency of rules and processes.
  - By law, a ship heading to a US port has to provide a cargo report to US Customs 24 hours before it sails.
Process / Rule Consistency

- Customer relationship management processes at IBM too complicated to verify manually.
- Represented company policies as constraints on business processes.
- Tested consistency with PSL translation of those processes.
- Identified ten problems, four of which had not been discovered by rollout.

Partial Process Specification

- PSL supports declaring as much or as little as needed about a process.
  - First order constraints on execution model.
- Turns ambiguity (unintentional omission) into abstraction (intentional omission).
  - Did the modeler intend that no other step occur between Paint and Dry?
- Many applications to process: categorization, search and matching, design management, protocols, rule / policy / process integrity.
More Information

- **PSL Specification:**

- **Introduction:**

- **NIST PSL site:**
  - [http://www.nist.gov/psl](http://www.nist.gov/psl)

- **Other material:**
  - [http://www.conradbock.org/#PSL](http://www.conradbock.org/#PSL)
  - conrad.bock at nist.gov