Manufacturing Engineering

PSL and Flow Models

Conrad Bock

Michael Gruninger

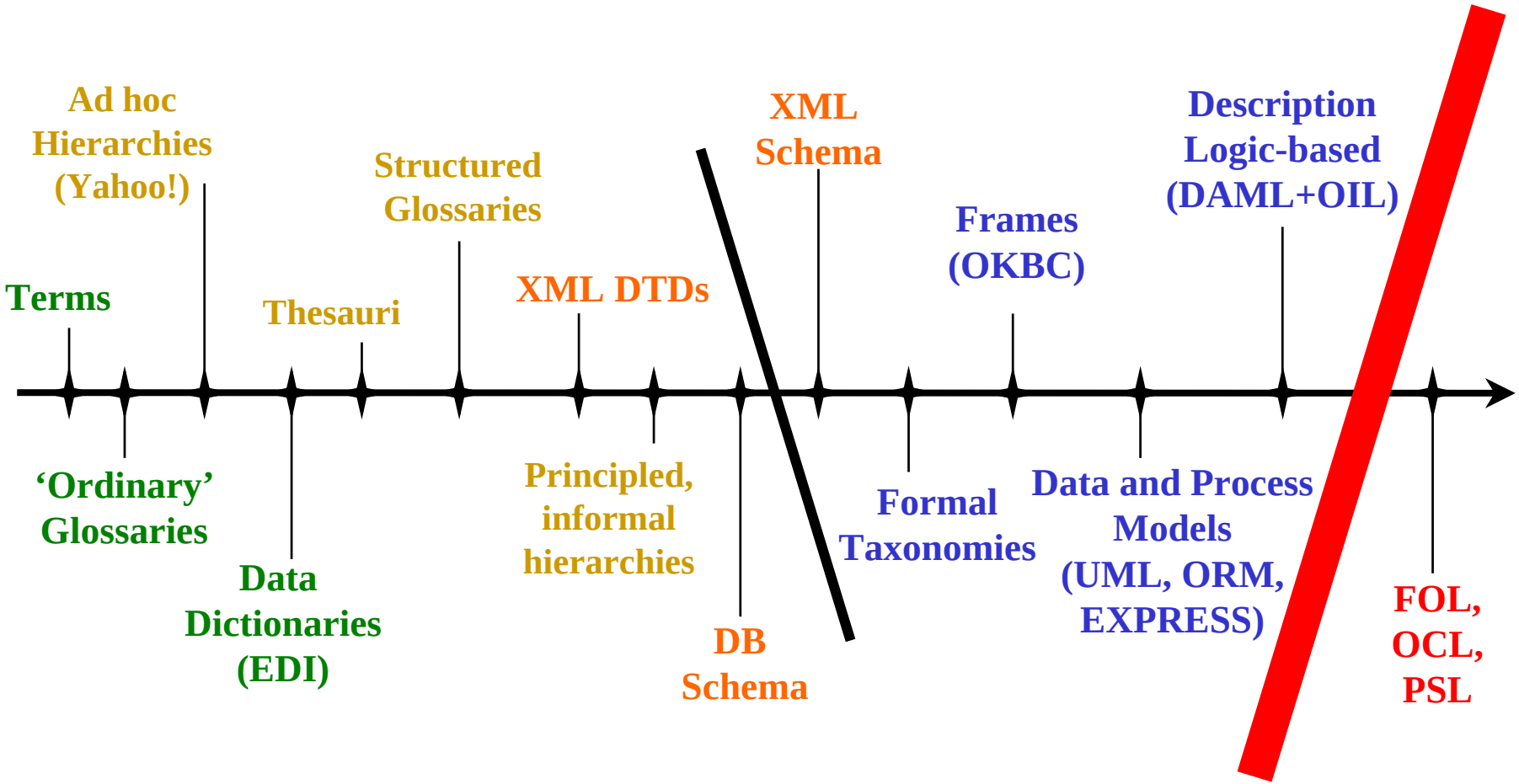
8/2004

NIST

Overview

- Approaches to system specification
  - Model vs instance-based
  - Example from structural specification
- PSL introduction
  - Why PSL is not yet another “L”.
  - Basic PSL concepts
  - How PSL is used
- PSL application
  - Behavior Classification
- Conclusions

Ontology Languages



Glossaries & Data Dictionaries

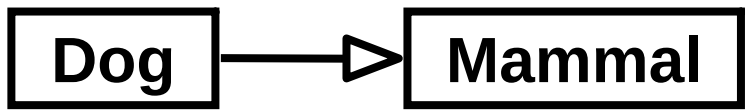
Thesauri, Taxonomies

Formal Languages & Automated Reasoning

Left of Red Line (User view)

- **OWL:**

```
<owl:Class rdf:ID="Mammal"/>
<owl:Class rdf:ID="Dog">
  <rdfs:subClassOf rdf:resource="#Mammal"/>
</owl:Class>
```

- **UML:** 

(or UML repository)

- **C++: `struct Dog : Mammal { }`**

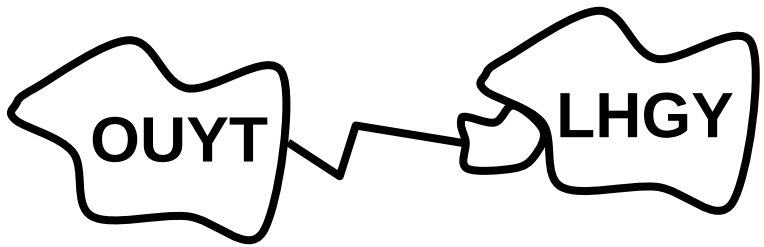
- **English: Dog is a kind of Mammal**

Left of Red Line (Machine view)

- **OWL:**

```
<owl:GWJK rdf:ID=“LHGY"/>
<owl: GWJK rdf:ID=“OUYT">
  <rdfs:LNCGWJKYO rdf:resource="#LHGY" />
</owl: GWJK>
```

- **UML:**



(same for repository)

- **C++: eghc OUYT : LHGY {}**
- **English: OUYT er a bfvc yo LHGY**

Specialized Interpreters

- **Interpreters built for each LORLL ...**
- **… by humans who "know" the meanings.**
- **"Consensus" achieved by:**
  - **Documentation, runtime examples, model theories, RORLL's.**
- **LORLL's are fundamentally:**
  - **Not self-documenting.**
  - **Don't say what they mean.**
- **Result: Interoperability problems.**

Right of Red Line

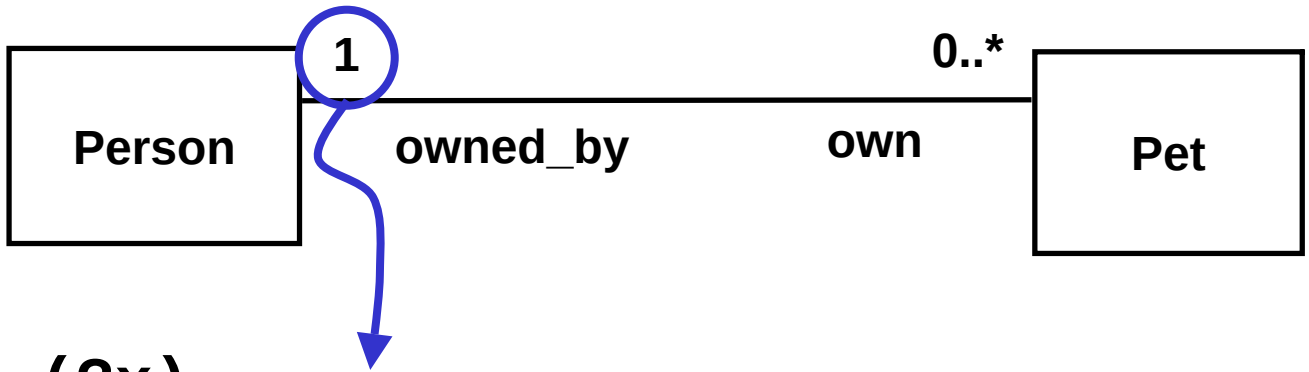
- **FOL:**

```
(forall (?x)
   (implies (Dog ?x)
            (Mammal ?x)))
```

- **Self documenting because it refers to *instances* of domain concepts (?x).**
- **Still need interpreter for “forall”, etc.**
- **Small set of highly reusable and composable constructs.**

Right of Red Line

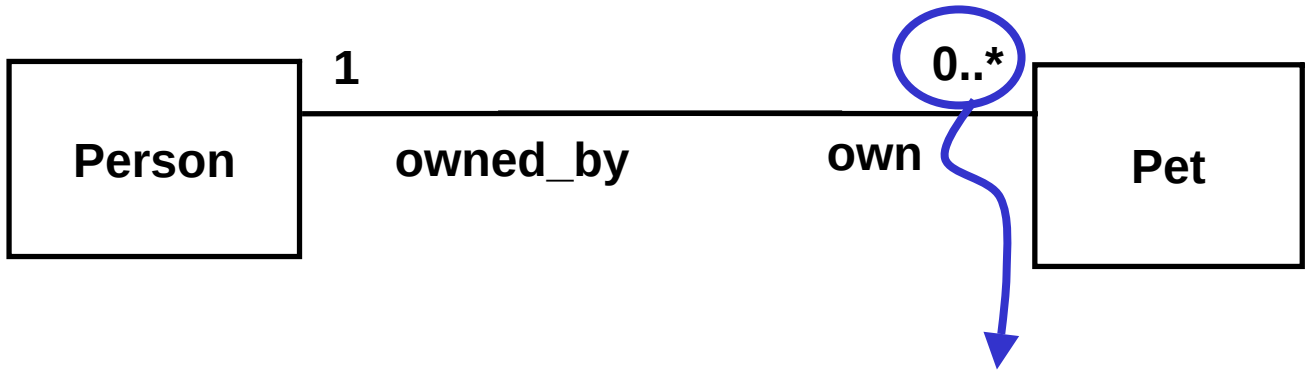
- **Simple things can be hard to say:**




```
(forall (?x)
   (implies (Pet ?x)
            (exists (?y)
               (and (Person ?y)
                    (own ?y ?x)
                    (forall (?z)
                       (implies (own ?z ?x)
                                (= ?z ?y)))))))
```

Right of Red Line

- **And some things impossible:**



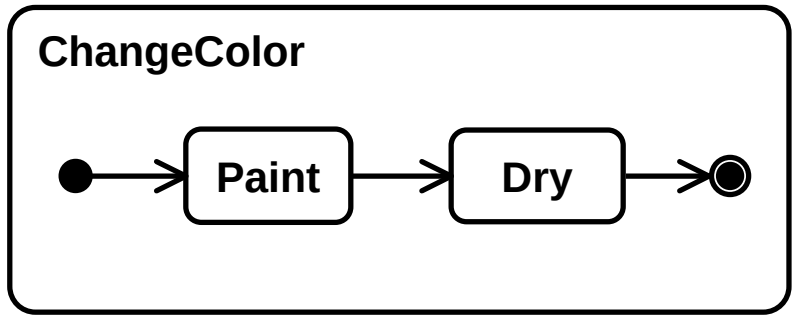

```
Each person owns:
                { 0 pets, or
                { or 1 pet,
                { or 2 pets
                { or 3 pets . .
Each one
expanding
to increasingly
complicated
expression
```

Left/Right Comparison

- **No silver bullet**
- **Left of Red Line (modeling):**
  - **Usually more concise.**
  - **Easier to add concepts.**
    - ***Except for updating tools*.**
  - **Difficult to interpret correctly.**
- **Right of Red Line (instance-based):**
  - **Self-documenting.**
  - **Sometimes very difficult to add concepts.**
    - ***Once done, tools understand the new concepts*.**
  - **Usually more verbose.**

Flow models: *LORLL*

- **UML 2:**



(or UML repository)

- **BPEL:**

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

- **C:**

```
void ChangeColor
{ Paint();
  Dry();
}
```

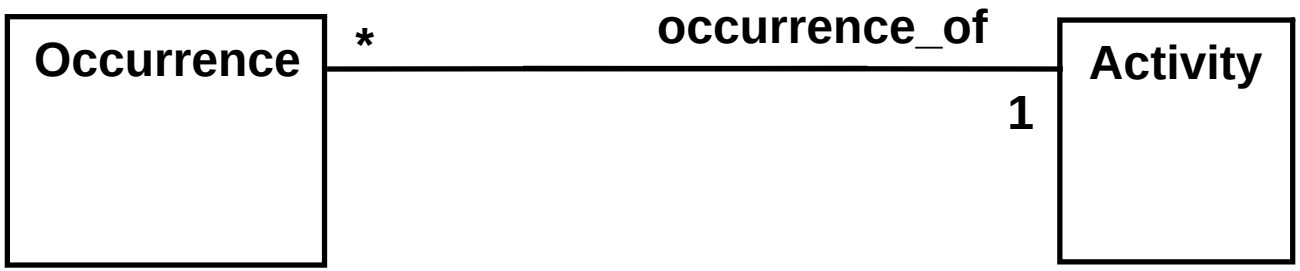
Specialized Interpreters

- **Interpretation is needed to know:**
  - **Can any other activities occur between Paint and Dry?**
  - **What behaviors can occur concurrently with painting?**
  - **How soon after painting must drying occur?**
  - **Is it possible under exceptional conditions for drying not to happen?**

PSL: *RORLL (instance-based)*

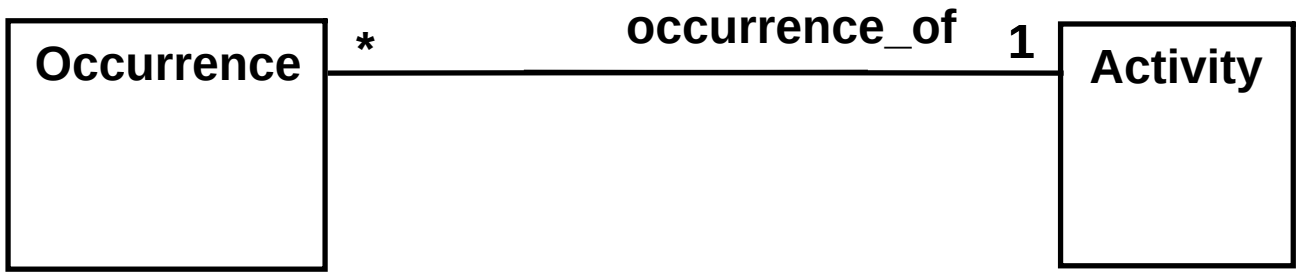
- **Instances of processes**
  - **Individual executing processes.**
  - **ChangeColor executed at 10:21am ET 9/1/2003 at factory 1.**
- **Execution sequence**
  - **Sequences of executing steps in the process, perhaps some concurrently.**
  - **Paint executed at 10:22am, then Dry at 10:40am, etc.**
- **Small set of highly reusable constructs.**

Basic PSL Concepts



- **Occurrence is an execution of an Activity**
  - **like Paint executed at 10:22am ET 9/1/2003 at factory 1.**
- **Activity is a RORL-like**
  - **like Paint or Dry.**

Basic PSL Concepts

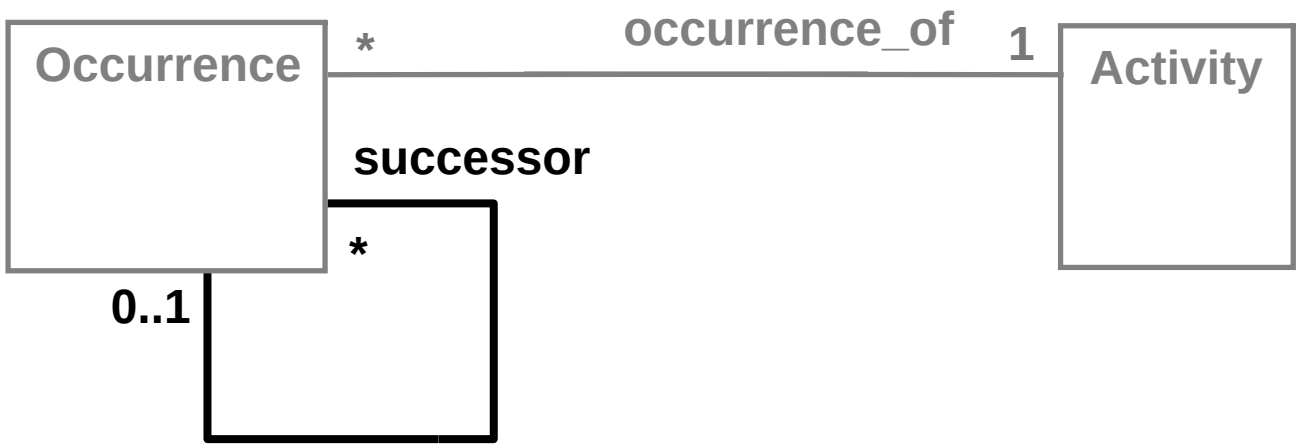


- **In FOL:**

```
(forall (?a ?occ)
   (implies (occurrence_of ?occ ?a)
            (and (activity ?a) (activity_occurrence ?occ))))
(forall (?occ)
   (implies (activity_occurrence ?occ)
            (exists (?a)
               (and (activity ?a) (occurrence_of ?occ ?a)))))
(forall (?occ ?a1 ?a2)
        (implies (and (occurrence_of ?occ ?a1)
                      (occurrence_of ?occ ?a2))
                 (equal ?a1 ?a2))))
```

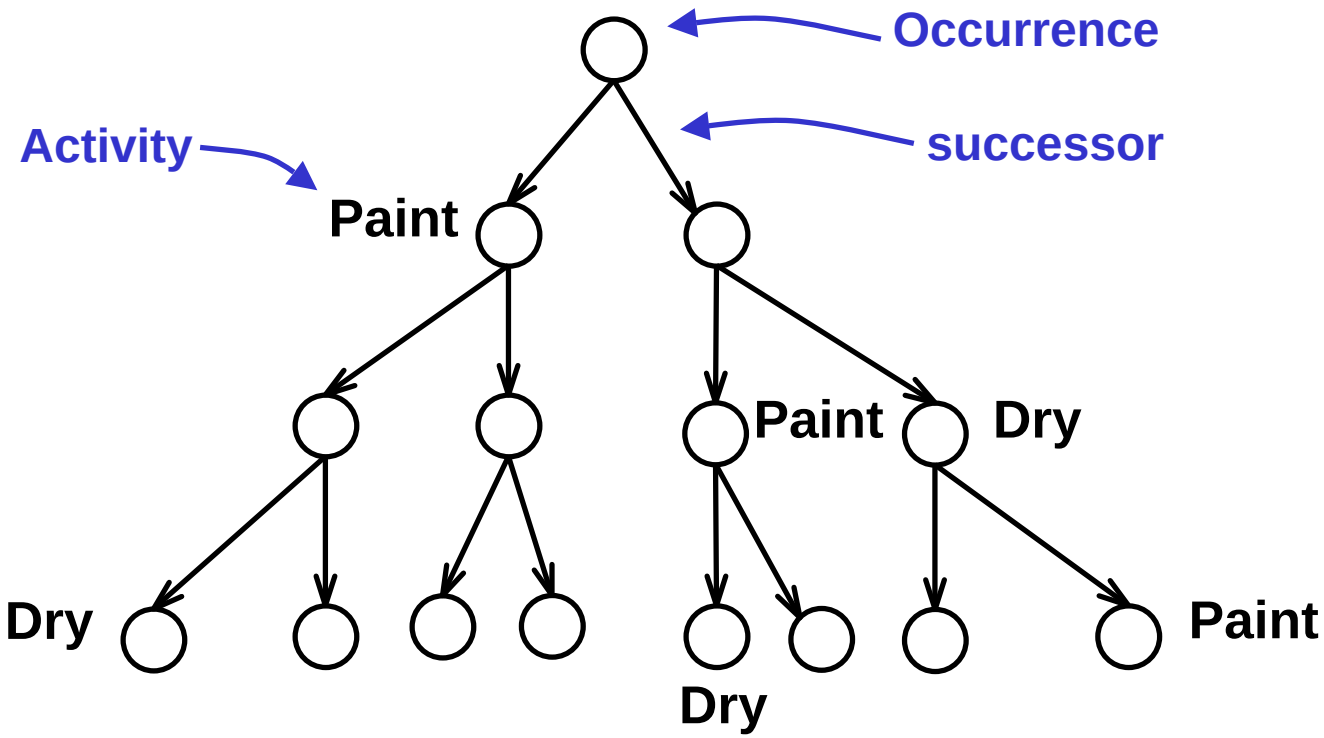
- **PSL is an execution-based way of describing processes.**
- **PSL happens to be expressed in FOL, but it is not bound to FOL.**

Basic PSL Concepts



- **Executions happen one after another.**
- **Covers all activities happening anywhere.**
- **Occurrence has multiple successors, one for each (theoretically) possible next occurrence.**

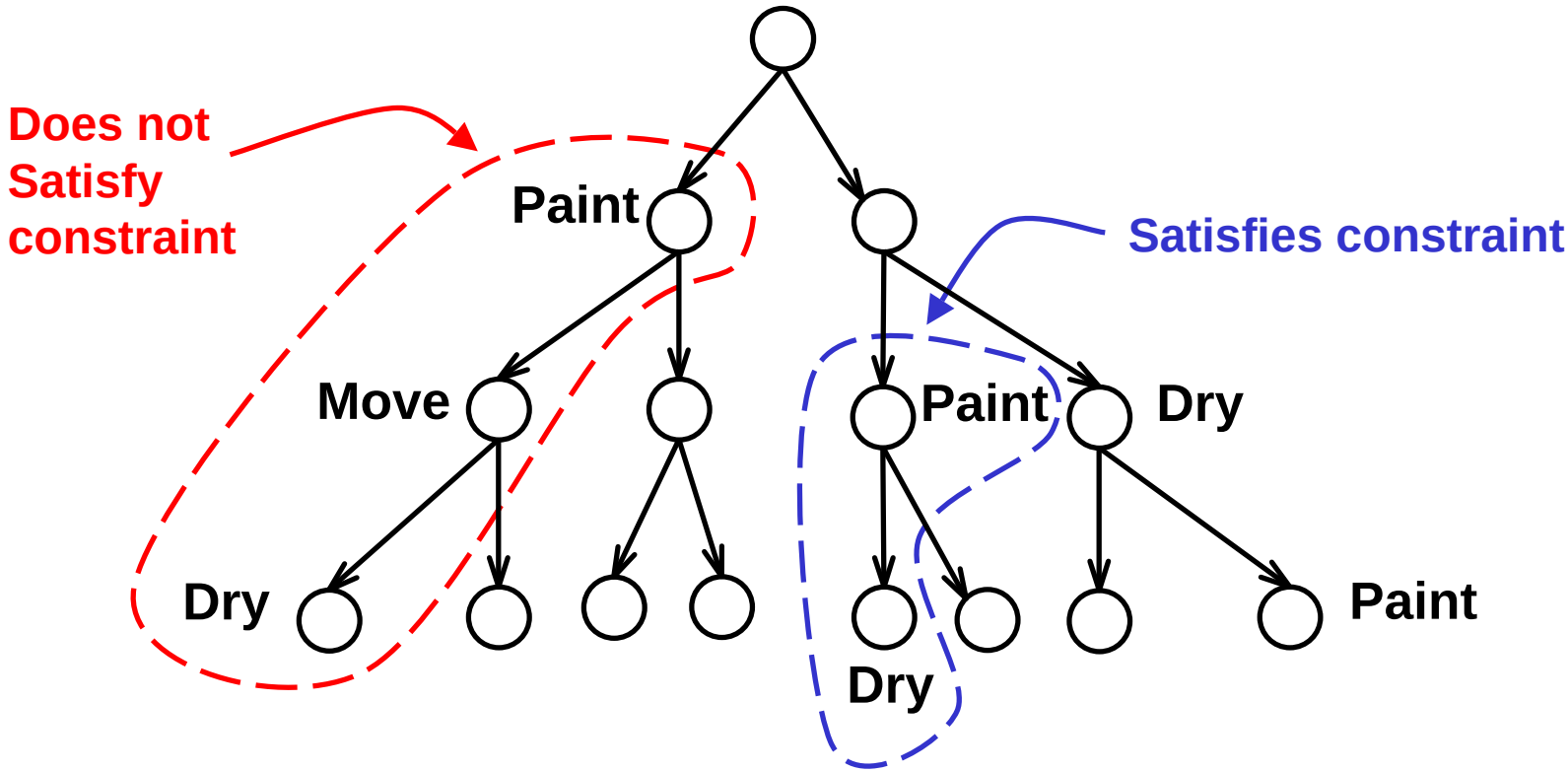
Occurrence Tree



- **Tree of all possible execution sequences, including those that**
  - **are not physically possible.**
  - **are not specified by the user.**
- **Not stored anywhere, just referred to.**

Process Specification in PSL

- **Constraints on the occurrence tree.**
- **Example: drying immediately follows all painting.**

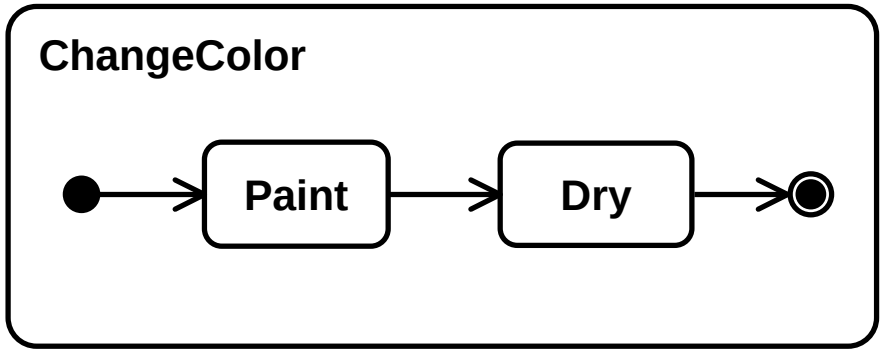


Process Specification in PSL

- **Constrain occurrences of Paint to be followed by occurrences of Dry:**

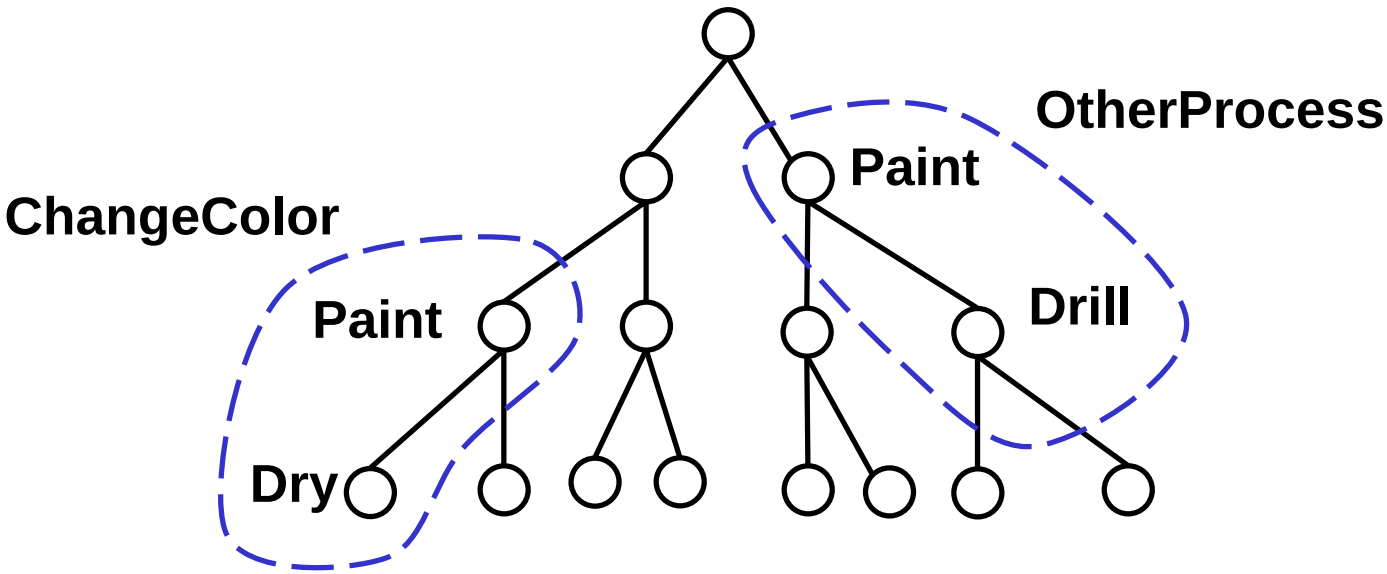
```
(forall (?occPaint)
  (implies
    (and (occurrence_of ?occPaint Paint)
         (legal ?occPaint))
    (and (legal (successor Dry ?occPaint))
         (forall (?otherSuccessor)
           (implies
             (not (equal ?otherSuccessor
                         (successor Dry ?occPaint)))
             (not (legal ?otherSuccessor)))))))
```

Process Specification in PSL



- **Above says that Dry happens after Paint *under executions of ChangeColor*.**
- **Other processes may use Paint without Dry.**

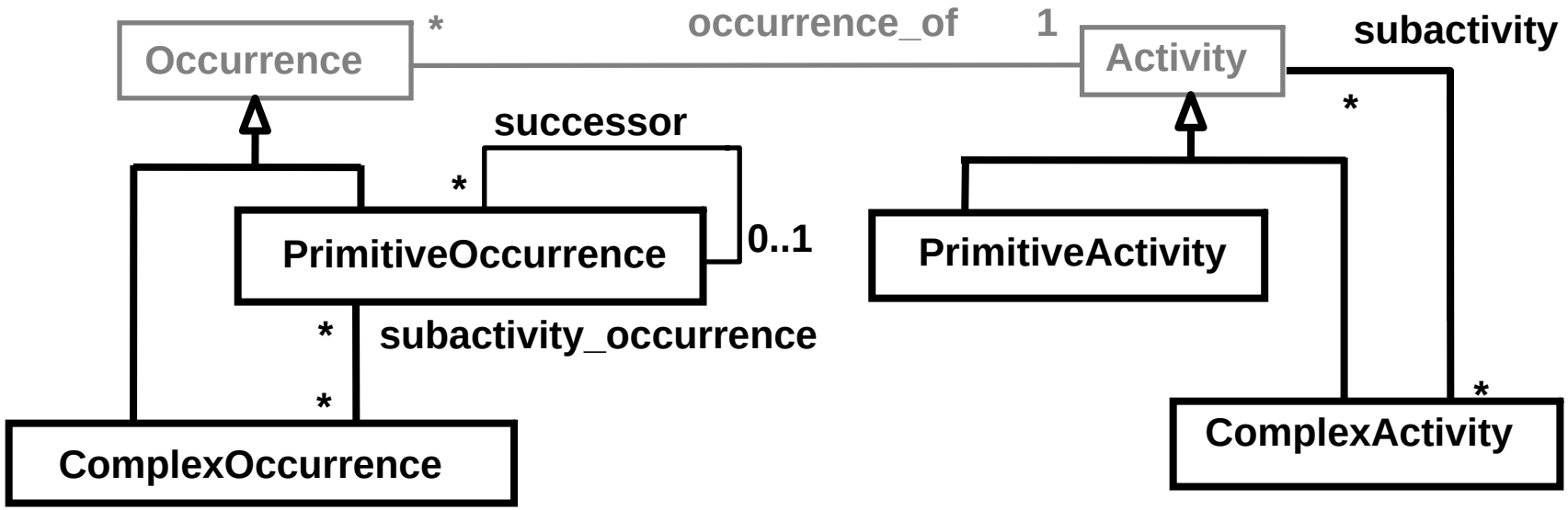
Complex Processes in PSL



- **Paint happens immediately after Dry under executions of ChangeColor.**
- **ChangeColor specification does not constrain OtherProcess above**

Complex Processes in PSL

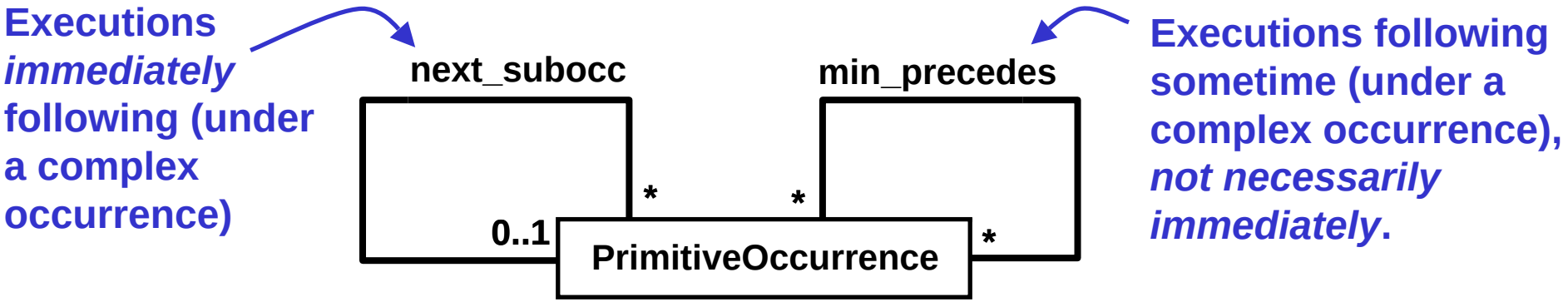
- **Complex occurrences and activities composed of primitive ones:**



- **Successor moved down to PrimitiveOccurrence.**
- **Occurrence tree covers every step at finest grain.**

Complex Processes in PSL

- **Execution sequencing within complex activity:**



- **min_precedes defined in terms of successor.**
- **next_subocc in terms of min_precedes:**

```
(forall (?s1 ?s2 ?s3)
   (iff (next_subocc ?s1 ?s2 ?a)
        (and (min_precedes ?s1 ?s2 ?a)
             (not (exists (?s3)
                  (and (min_precedes ?s1 ?s3 ?a)
                       (min_precedes ?s3 ?s2 ?a))))))
```

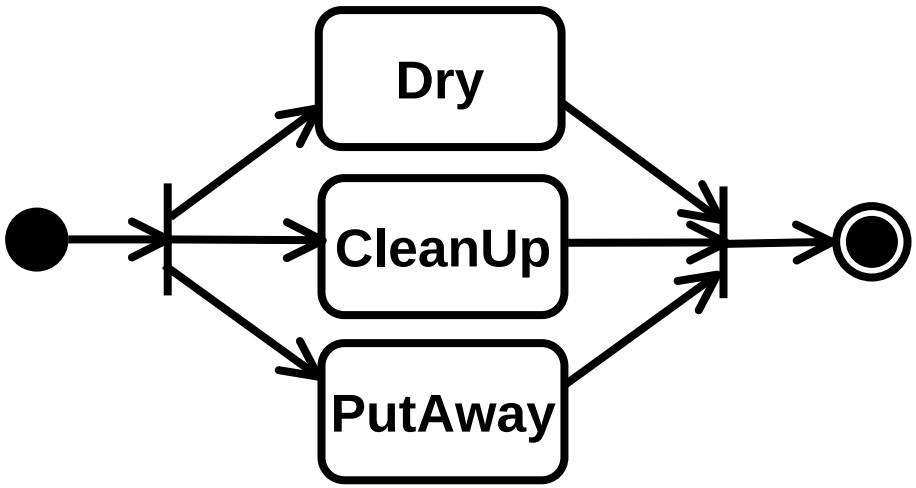
Complex Processes in PSL

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

```
(forall (?occChangeColor)
  (implies
    (occurrence_of ?occChangeColor ChangeColor)
    (exists (?occPaint ?occDry)
      (and (occurrence_of ?occPaint Paint)
           (occurrence_of ?occDry Dry)
           (subactivity_occurrence ?occPaint ?occChangeColor
           (subactivity_occurrence ?occDry ?occChangeColor)
           (next_subocc ?occPaint ?occDry
                        ChangeColor)))))
```

Process Specification in PSL

- **Simple things can be hard to say:**



6 nonoverlapping orderings

6 partially overlapping orderings

1 complete overlapping order

Process Specification in PSL

- **Benefits:**
  - **Self-documenting (says what it means).**
  - **Small set of highly reusable concepts.**
  - **Improved interoperability by reducing ambiguity.**
- **Disadvantages**
  - **Sometimes difficult to add concepts.**
  - **More verbose in many cases.**
- **Additional benefit to process modeling:**
  - **More flexible constraints (classification, rules).**

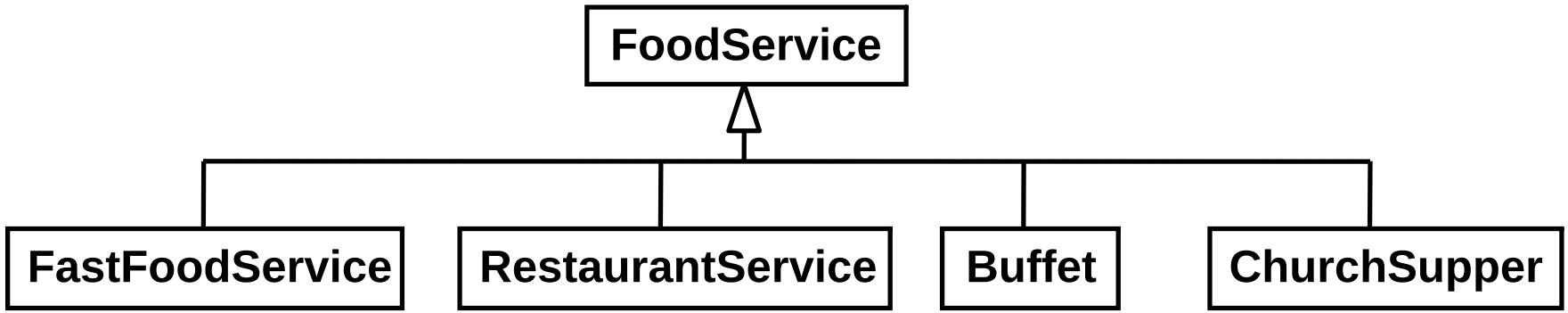
Advertise the Distinction

- **Common to think of PSL as yet another “L” (UML, BPEL, etc).**
- **PSL is a semantic foundation for all LORR flow/process models.**
- **Even KBSI substitutes flow models for PSL (PDS).**
- **More expressive and less ambiguous than flow models.**

How to Get Best of Both Worlds?

- **Research topic**
- **Translate models to instance-based**
  - **Not enough: Users ignore instance-based**
- **Instance-based aid to example testing**
  - **Check examples (user-defined or actual) against instance-based semantics.**
  - **Generate examples from instance-based specs to be checked by users or system.**
- **Annotate modeling languages with instance-semantics.**

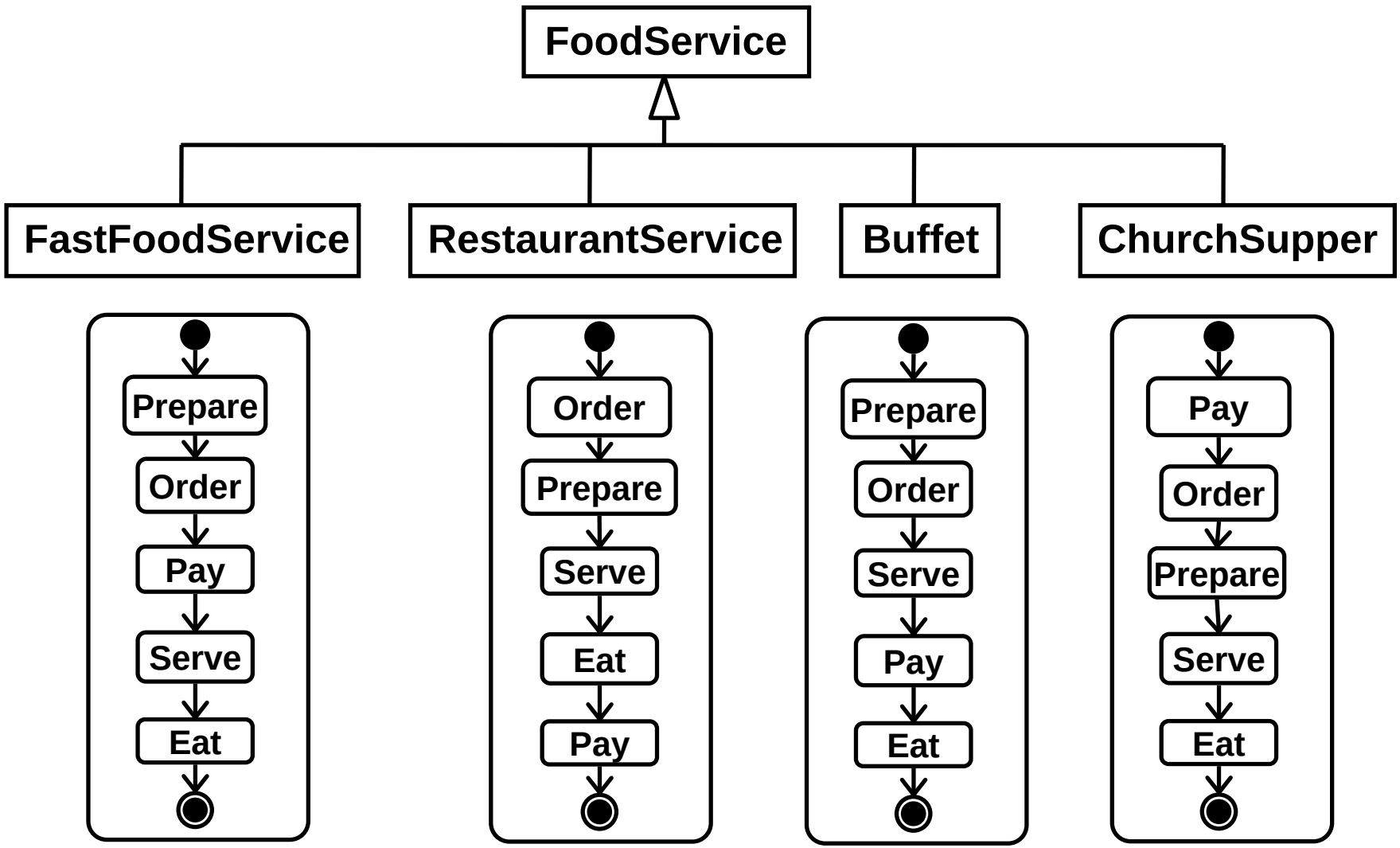
Behavior Classification



- **Classification of process executions:**

```
(forall (?occFFS)
  (implies (occurrence_of ?occFFS FastFoodService)
           (exists (?occFS)
             (and (occurrence_of ?occFS ?FoodService)
                  (forall (?s)
                   (implies
                     (subactivity_occurrence ?s ?occFFS)
                     (subactivity_occurrence ?s ?occFS)))))))
```

Behavior Classification



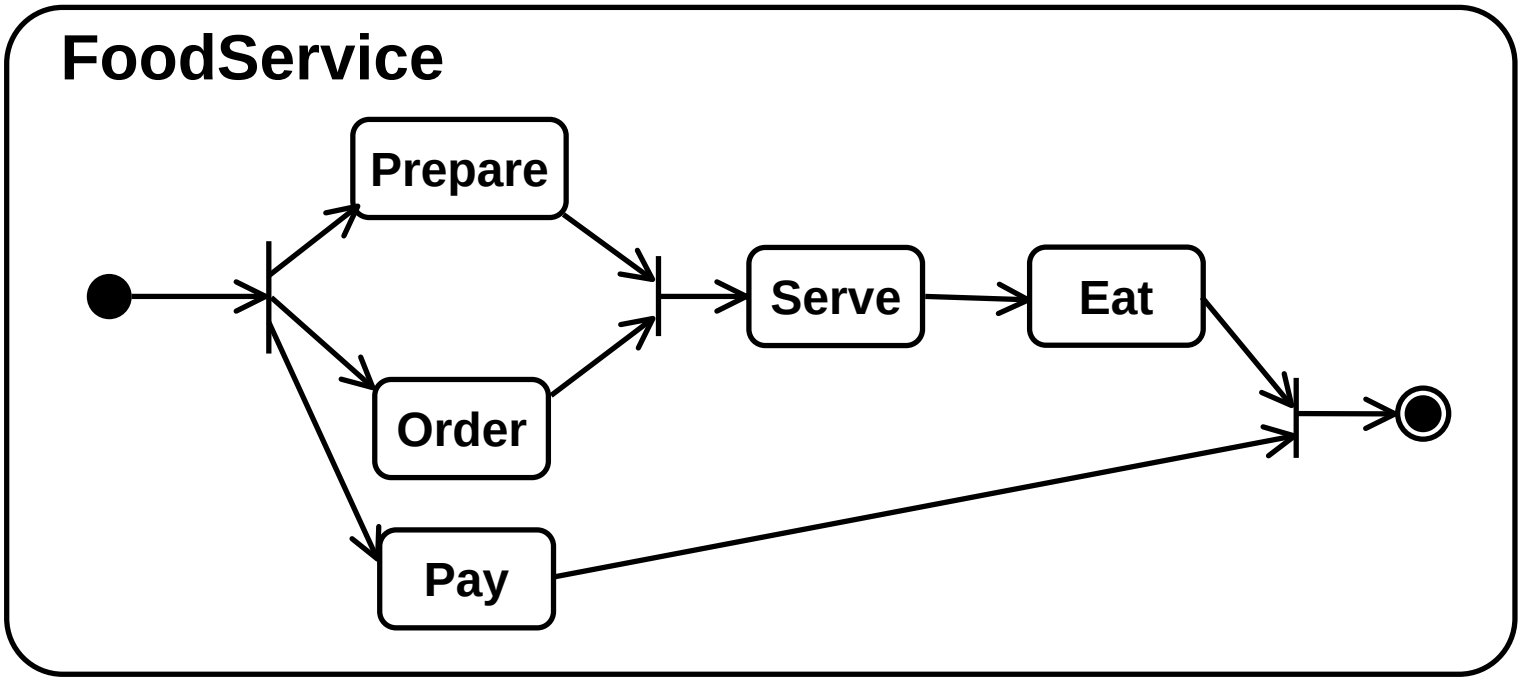
- How to abstract commonality?

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.**
- **Need to *partially* specify a process as incrementally-defined constraints.**

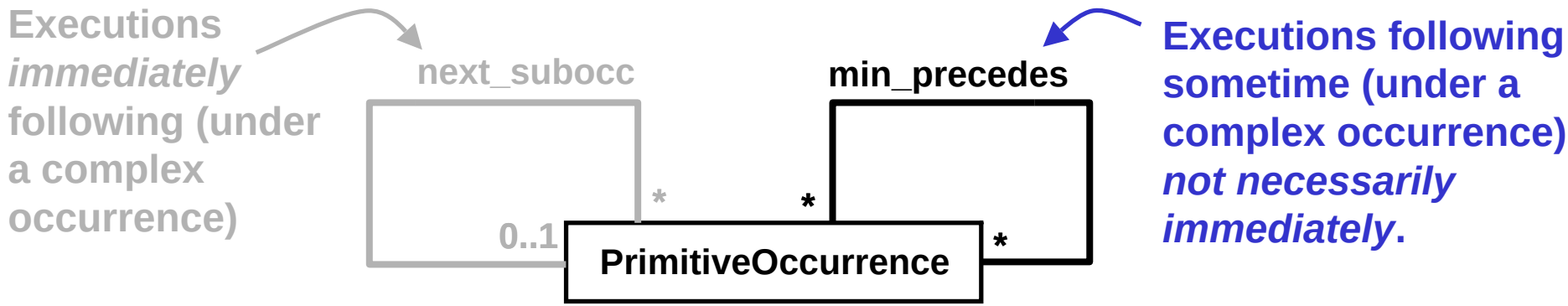
Behavior Classification

- **Flow models are not expressive enough:**



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

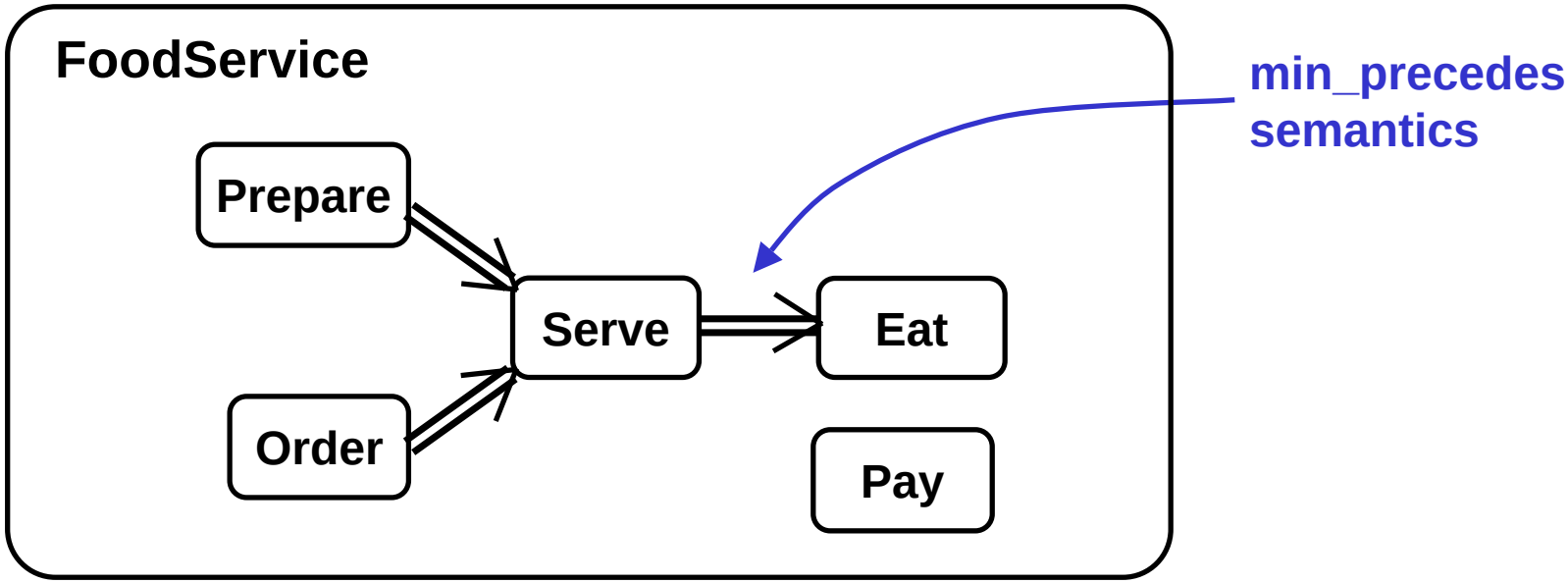
Behavior Classification



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

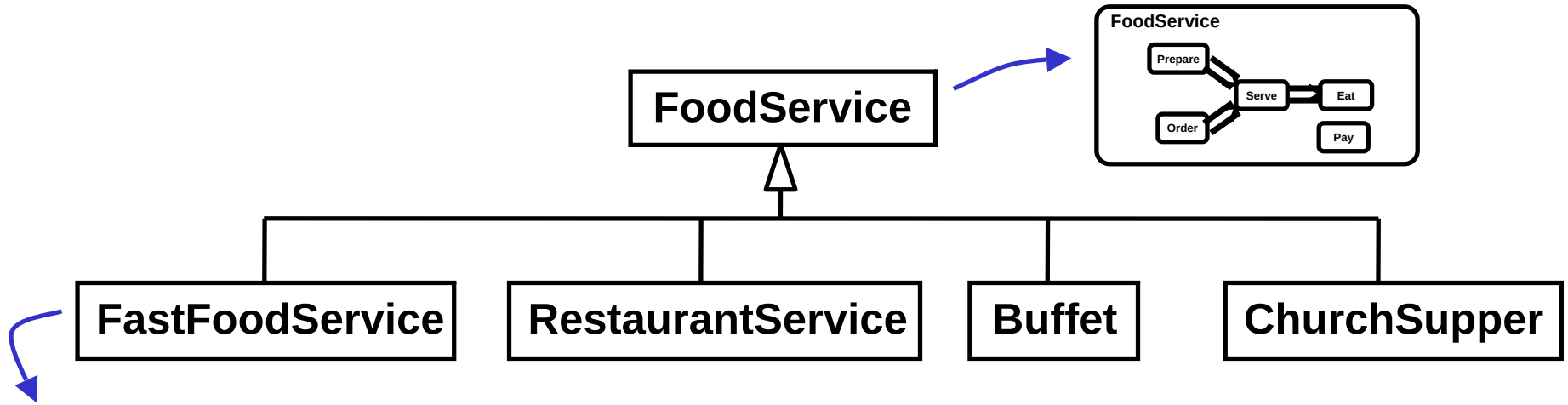
```
(forall (?occFoodService)
  (implies
    (occurrence_of ?occFoodService FoodService)
    (exists (?occPrepare ?occEat)
      (and
        (occurrence_of ?occPrepare Prepare)
        (occurrence_of ?occEat Eat)
        (subactivity_occurrence ?occPrepare ?occFoodService)
        (subactivity_occurrence ?occServe ?occFoodService)
        (min_precedes ?occPrepare ?occEat
                      FoodService)))))
```

Behavior Classification



- **Possible enhancement to UML notation.**

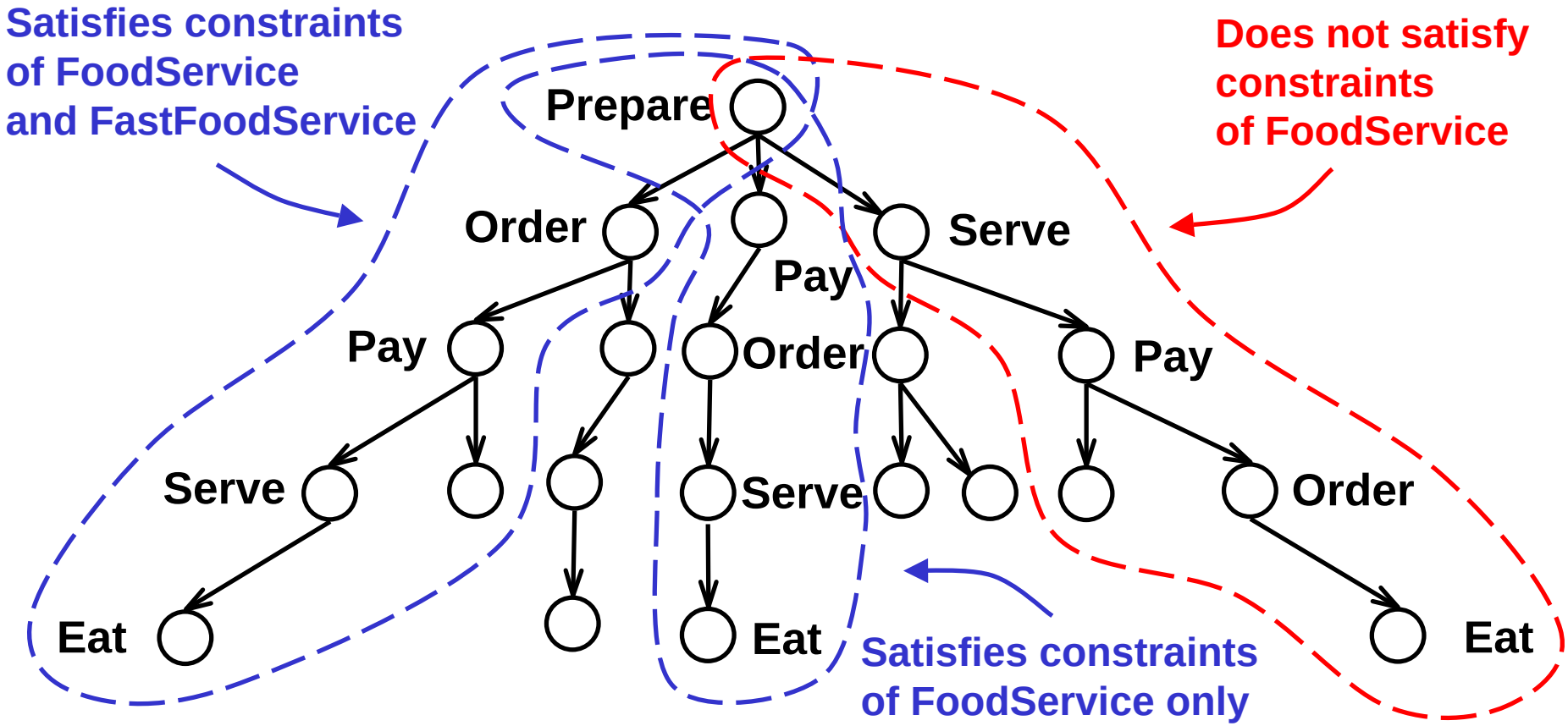
Behavior Classification



- **FastFoodService: Prepare sometime before Order**

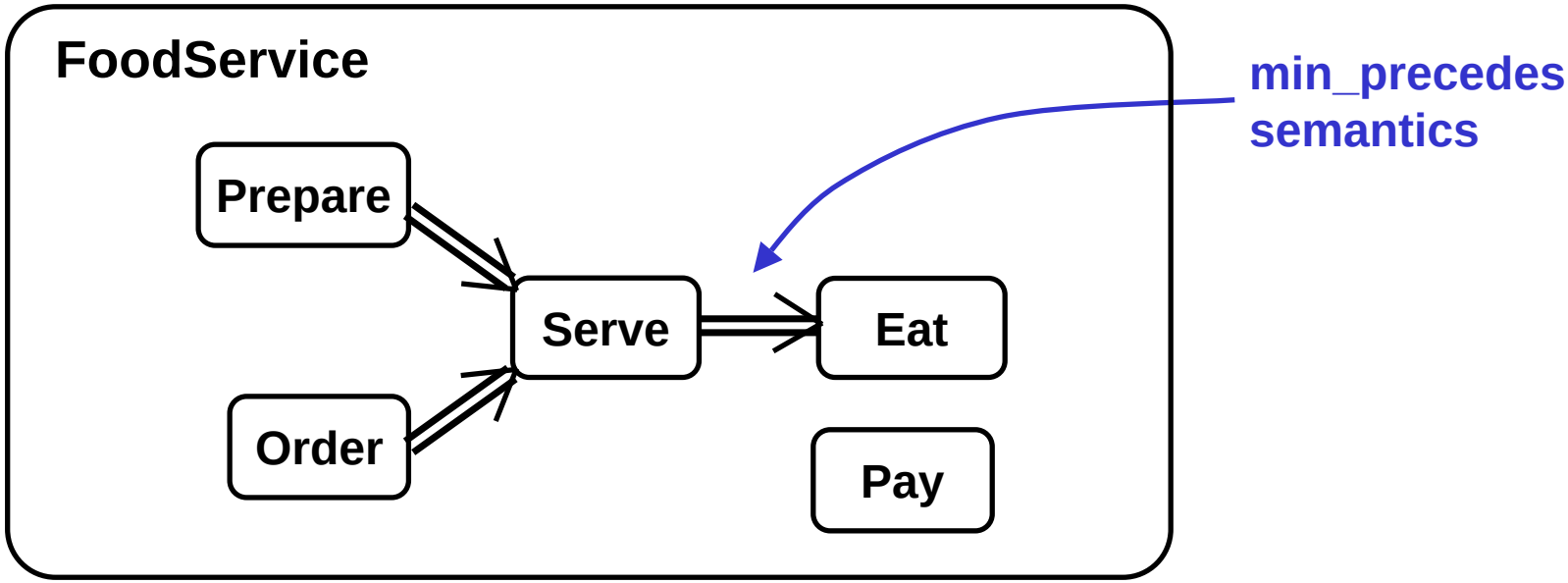
```
(forall (?occFastFoodService)
  (implies
    (occurrence_of ?occFastFoodService FastFoodService)
    (exists (?occPrepare ?occOrder)
      (and
        (occurrence_of ?occPrepare Prepare)
        (occurrence_of ?occOrder Order)
        (subactivity_occurrence ?occPrepare ?occFoodService)
        (subactivity_occurrence ?occOrder ?occFoodService)
        (min_precedes ?occPrepare ?occOrder
                      FoodService)))))
```

Behavior Classification



- **Execution traces classified by process specifications (constraints).**

Behavior Classification



- **Possible enhancement to UML notation.**
- **Requires updating tools and services.**

Abstraction vs Ambiguity

- **Both omit information.**
- **One does it intentionally and explicitly, the other doesn't.**
- **Example:**
  - **Did the modeler intend that no other step occur between Paint and Dry?**
  - **Design intent is lost.**
- **A proper abstraction would say what the modeler actually meant.**
- **PSL does this with the occurrence tree.**

PSL Myths

- Too precise
  - Can write “partial programs”
  - Can make useful distinctions
    - Weak and strong ordering
    - Weak and strong concurrency
    - Activity viewpoints
    - Occurrence, activity, activity class
  - Distinctions provide power
- Can’t say everything
  - Some things too complicated

More Information

- **See paper “PSL: A Semantic Domain for Flow Models”.**
- **More applications of PSL to flow modeling.**
- **Parameterized activities.**
- **Inputs and outputs, see NISTIR.**
- **Concurrency and external activities.**
- **Decision points/merges.**
- **Closure.**