

Automatic combination of (business) processes

Institute of Applied Informatics and Formal Description
Methods,
University of Karlsruhe

Workshop Semantic Help, Sixth International EGOV Conference
2007

Agenda

- **Motivation**
- **Petri nets**
- **Ontology**
- **Petri net Ontology**
- **Composition**
- **Algorithm**
- **Example**
- **Summary & Future Prospects**

Motivation

- **Semantic Help emerge from:**
 - Support of the user by finding of the correct processes respectively parts of processes
 - (proactive) support of the user by the combination of processes
- **Semantic is necessary to find the appropriate processes**

Petri nets

- **Petri net** were invented in 1962 by Carl Adam Petri
- A **Petri net** is one of several mathematical representations of discrete distributed systems
- As a modeling language, it graphically depicts the structure of a distributed system as a **directed bipartite graph** with annotations
- It can also be formally represented
- A **Petri net** consist of:
 - place nodes
 - transition nodes
 - direct arcs (connecting places with transitions)

Ontology

$$O = \{C, H_c, R, rel, A_o\}$$

- **C** ... is a set of concepts
- **H_c** ... is a concept-hierarchy
- **R** ... is the set of relations
- **rel** ... describes the function
- **A_o** ... is the set of axioms

Petri Net Ontology I

- **Definition of Petri net elements as OWL concepts:**
 - $C = \{\text{PetriNet}, \text{Place}, \text{Transition}, \text{FromPlace}, \text{ToPlace}\}$
- **Order of this concept in a taxonomy (hierachy):**
 - $H_C = \{(\text{PetriNet}, \text{Place}), (\text{PetriNet}, \text{Transition}), (\text{PetriNet}, \text{FromPlace}), (\text{PetriNet}, \text{ToPlace})\}$
- **Relations (properties):**
 - $R = \{\text{hasNode}, \text{hasArc}, \text{hasMarking}, \text{placeRef}, \text{transRef}\}$

Petri Net Ontology II

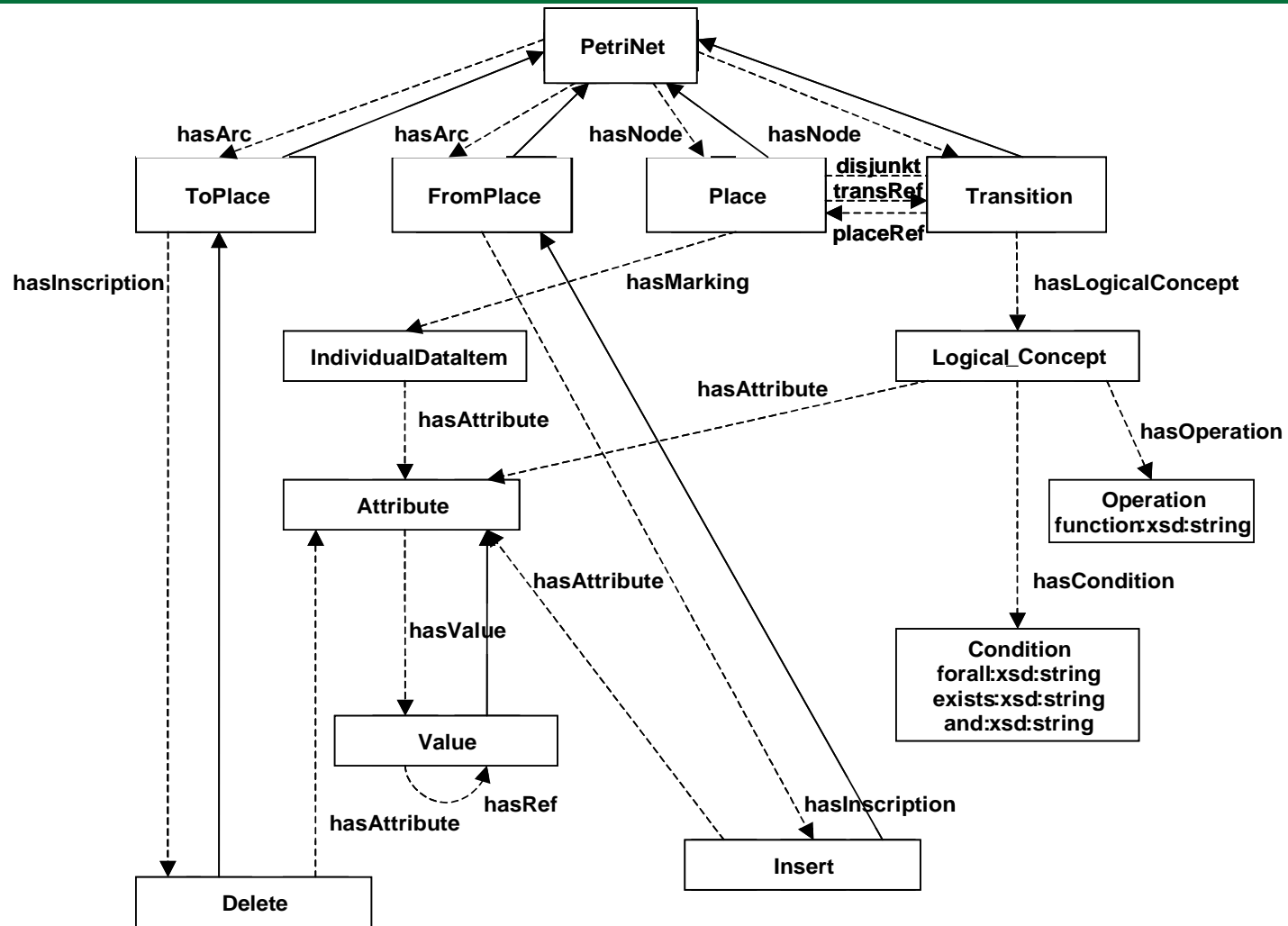
- **Domain and the values range of the relations**

- $rel = \{hasNode(PetriNet, Place), hasNode(PetriNet, Transition), hasArc(PetriNet, FromPlace), hasArc(PetriNet, ToPlace), hasMarking(Place), placeRef(Transition, Place), transRef(Place, Transition)\}$

- **Axiom**

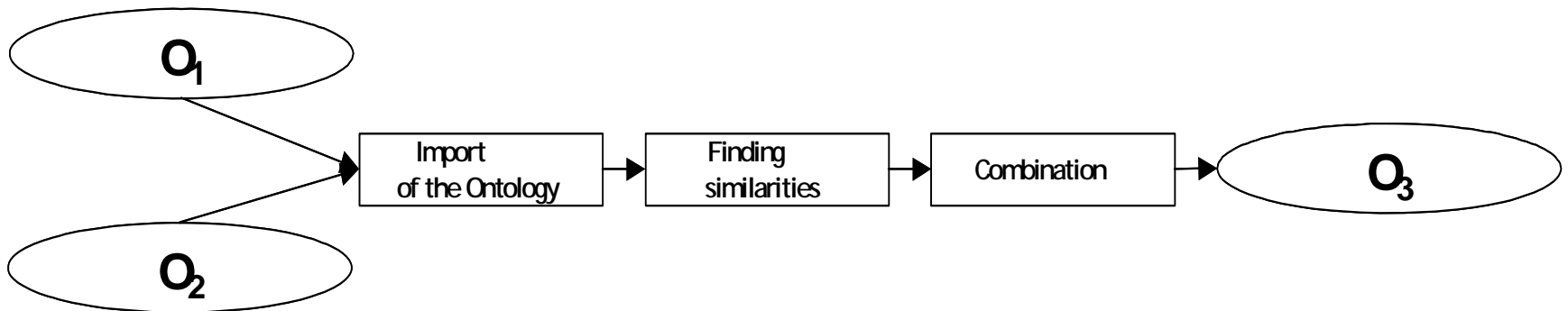
- $A_0 = \{(PetriNet \equiv \geq 1hasNode.(Transition \sqcup Place) \sqcap hasArc.(FromPlace \sqcup ToPlace) \sqcap Transition \equiv PlaceRef.Place \sqcap Place \equiv TransRef.Transition \sqcap = 1hasMarking \sqcap FromPlace \equiv 1hasNode.Place \sqcap ToPlace \equiv 1hasNode.Transition)\}$

Petri Net Ontology III



Approach to combine Petri Net Ontologies

1. Import of the Ontology in a tool, which detects similarities
2. Finding of semantically corresponding elements
3. Combination of the Ontologies O_1 & $O_2 \rightarrow O_3$
checking of the consistency, coherence, redundancy of O_3



Algorithm to combine Petri Net Ontologies I

1. Create the **head** of an Ontology
2. **Conflict test**
3. As long as **instances of nodes** are identically
 - a. Joined instances of nodes
 - b. Disjoined instances of nodes
4. The **instances of attributes and the instances of values** which are synonyms are getting joined together and get a common name
5. Create an **Input-node** in which the values from both Ontologies will be defined, also this attributes that do not occur in this site but their values are aligned with the object property `hasRef`

Algorithm to combine Petri Net Ontologies II

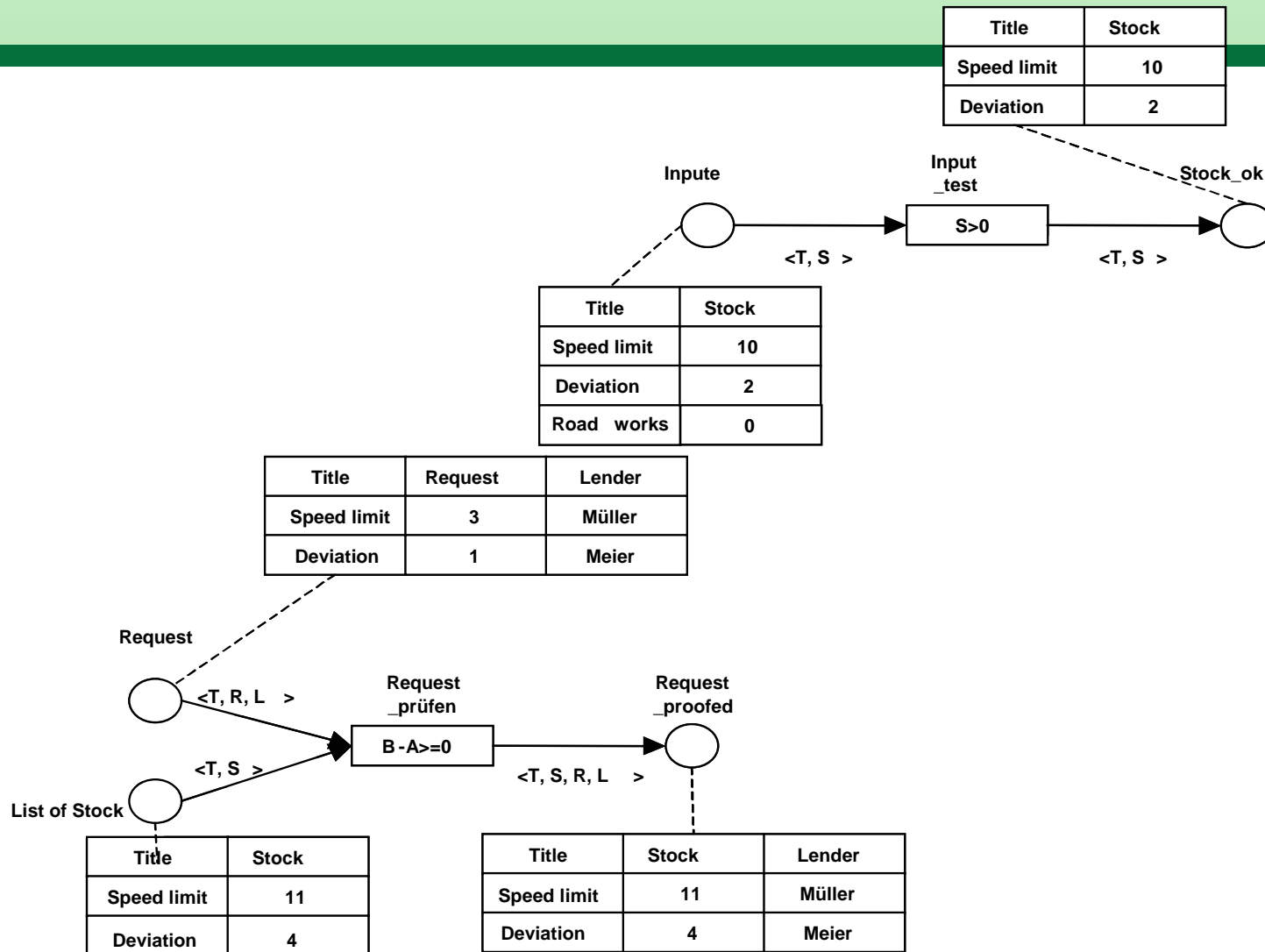
Concerning point 5: Relation of the attributes and values that should be merged

- a.** All attributes and values are identically
- b.** All attributes but not (all) values are identically
- c.** All values are identically, but there exist one (or several) additional values
- d.** Neither all values nor all attributes are identically

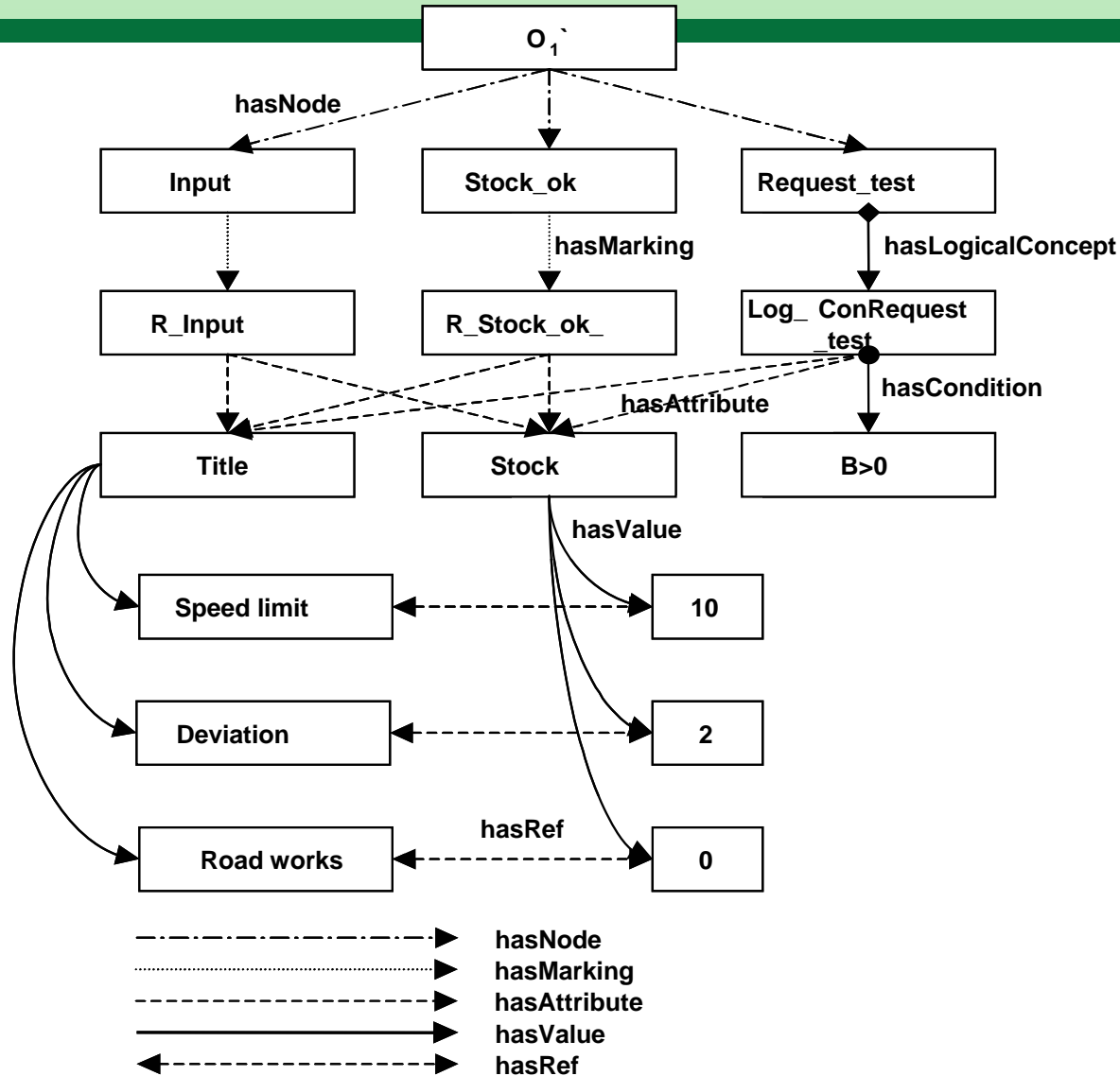
Algorithm to combine Petri Net Ontologies III

6. If a transition is to be joined, decide how the conditions should be joined together
7. Create new instances of node(s) from the merging instances of nodes of the Ontology O_3
8. Copy the remaining nodes and their reference in the Ontology O_3
9. Create an instance of the class `PetriNet` and associate the notes `hasNode` and the arcs `hasArc`
10. Check the consistency, coherence, redundancy of O_3

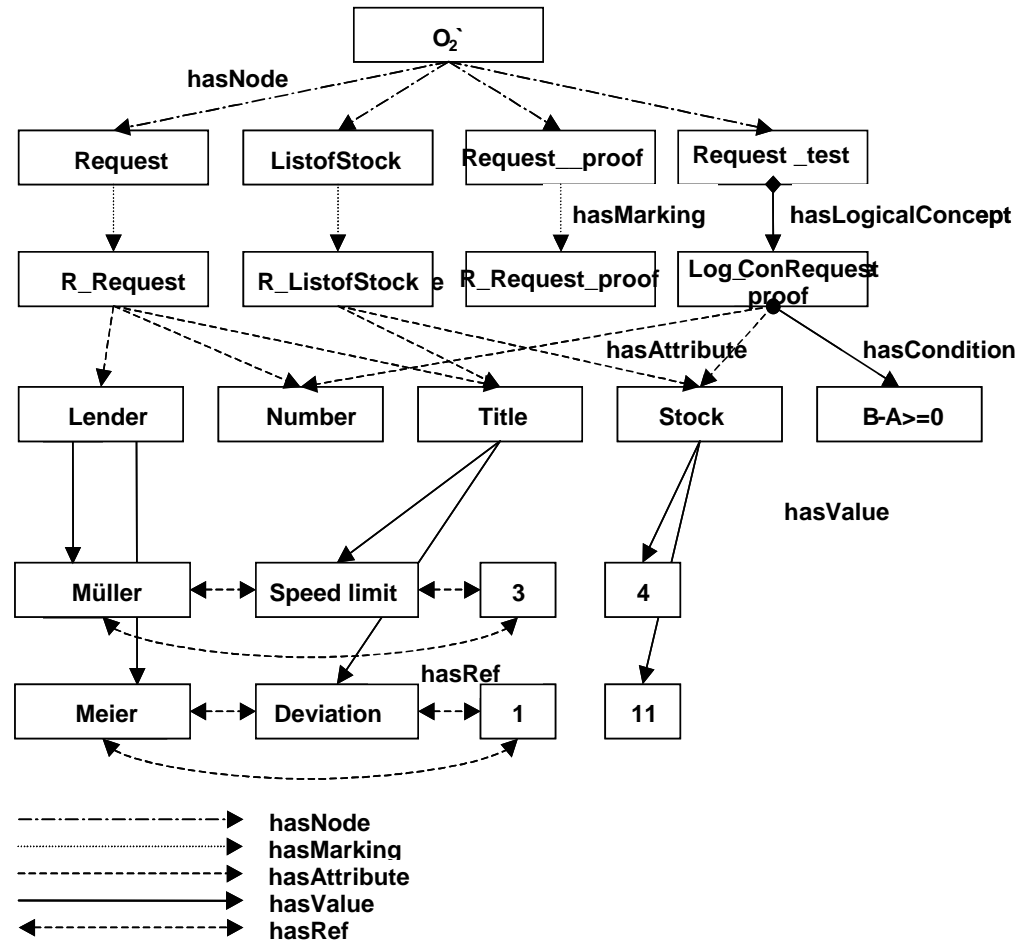
Example: 2 Petri nets



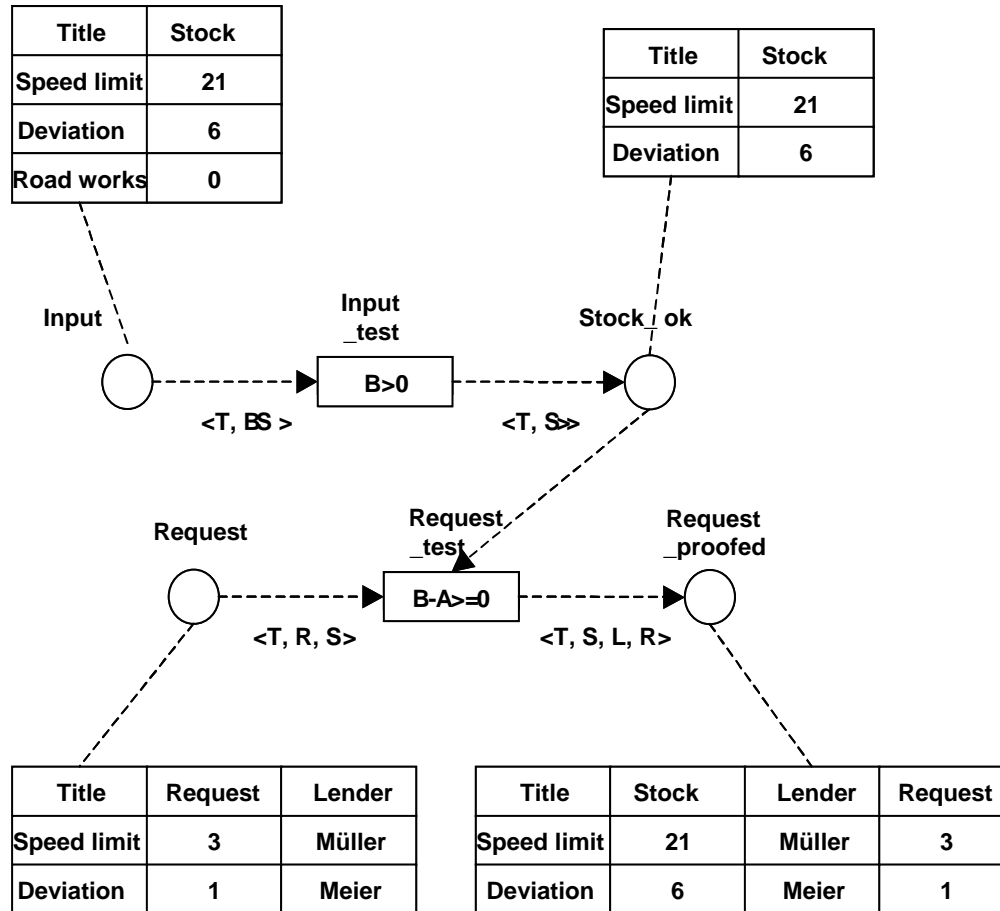
Example: Petri Net Ontology



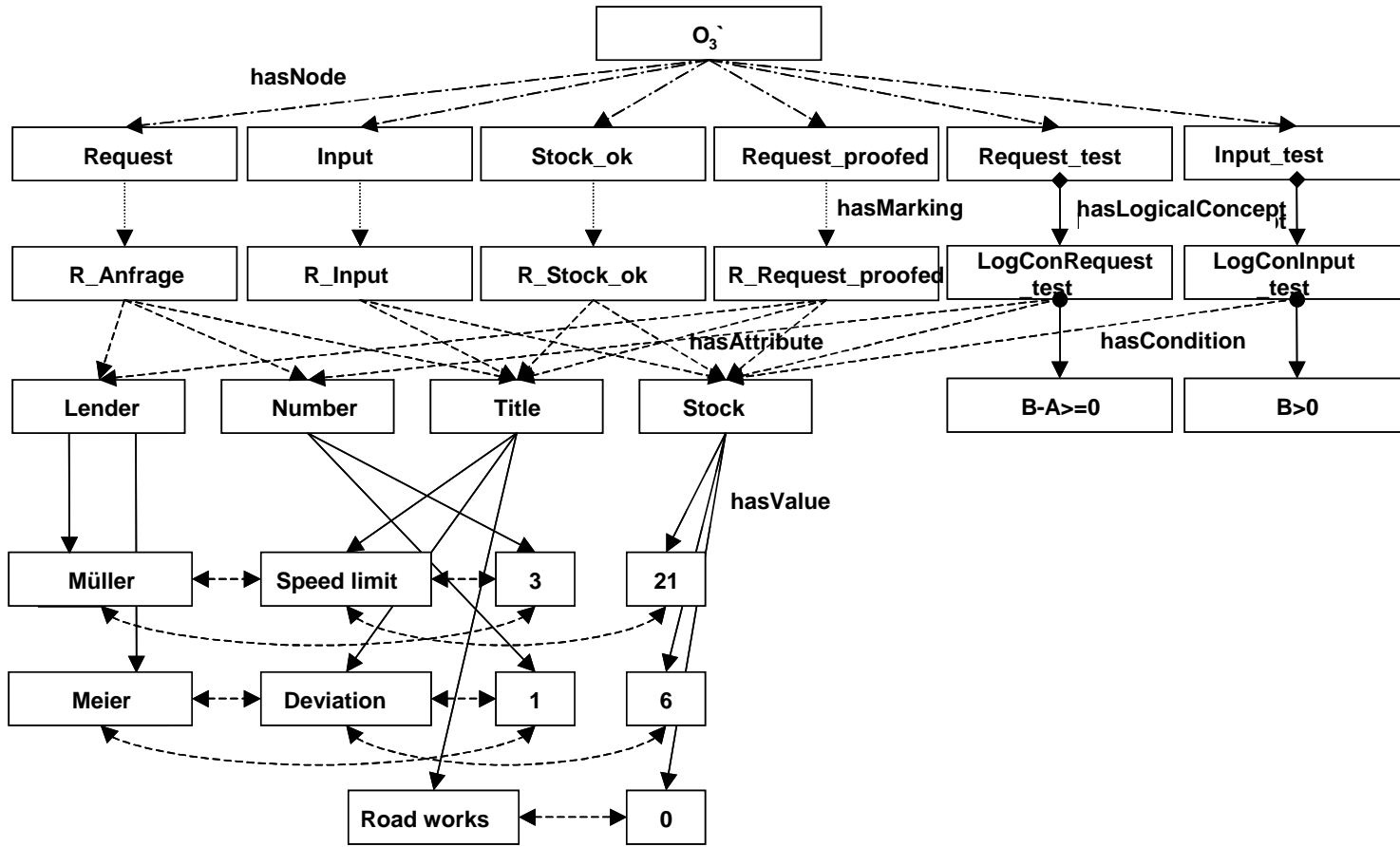
Example: Petri Net Ontology

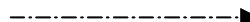
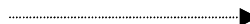
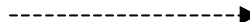

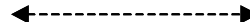


Example: Combined Petri net



Example: Petri Net Ontology



-  hasNode
-  hasMarking
-  hasAttribute
-  hasValue
-  hasRef

Summary and future prospects

- **Summary:**
 - The procedure of the combination can be automated
 - User interaction is necessary because a 100% match of the instances of the ontology is not present
- **Further questions:**
 - Level of abstraction of the processes
 - Change management
 - Data protection

Contact

- **Institut AIFB**
Universität Karlsruhe (TH)
76128 Karlsruhe
- Homepage:
www.aifb.uni-karlsruhe.de/BIK
- **Stefanie Betz**
stefanie.betz@aifb.uni-karlsruhe.de
- Telefon:
0721 / 608 – 4553

