D-PROV: Extending the PROV Provenance Model to express Workflow Structure

Paolo Missier[1], Saumen Dey[2], Khalid Belhajjame[3],
Vıctor Cuevas-Vicenttín[2], Bertram Ludaescher[2]

(1) Newcastle University, UK
(2) UC Davis, CA, USA
(3) University of Manchester, UK

TAPP'13
Lombard, IL.
April, 2013

DataONE

Queries that require provenance

Q1: “track the lineage of the final outputs of the workflow”

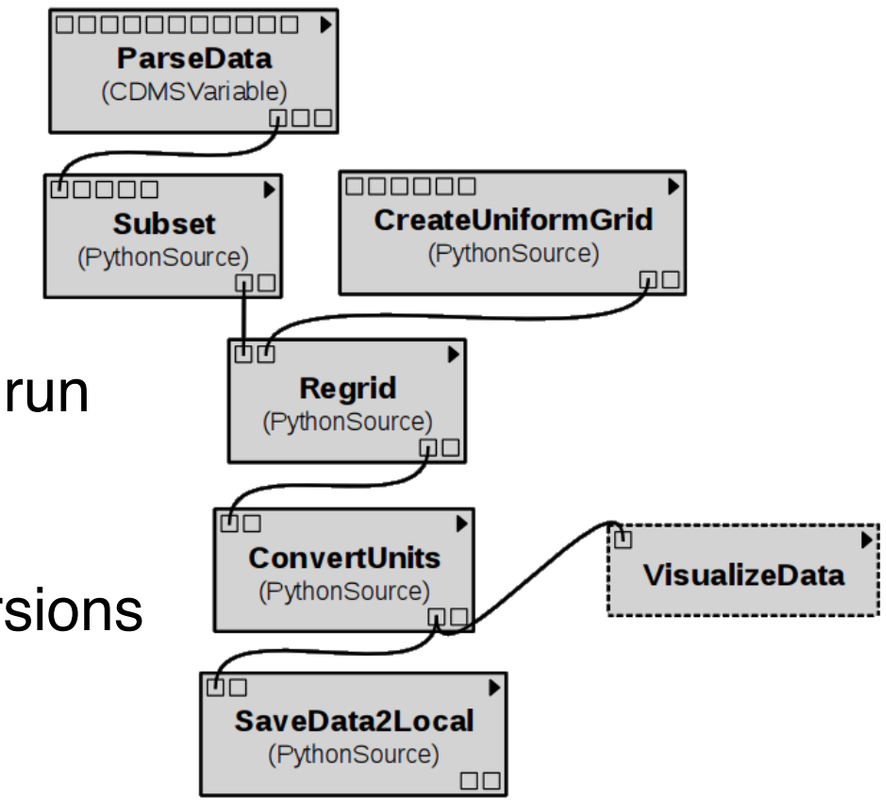
Q2: “list the parameter values that were used for a specific task t in the workflow”

Q3: “check that the provenance traces conform to the structure of the workflow”

Prospective Provenance (p-prov):
- representation of the workflow itself;

Retrospective Provenance (r-prov):
- provenance of the data produced by one workflow run

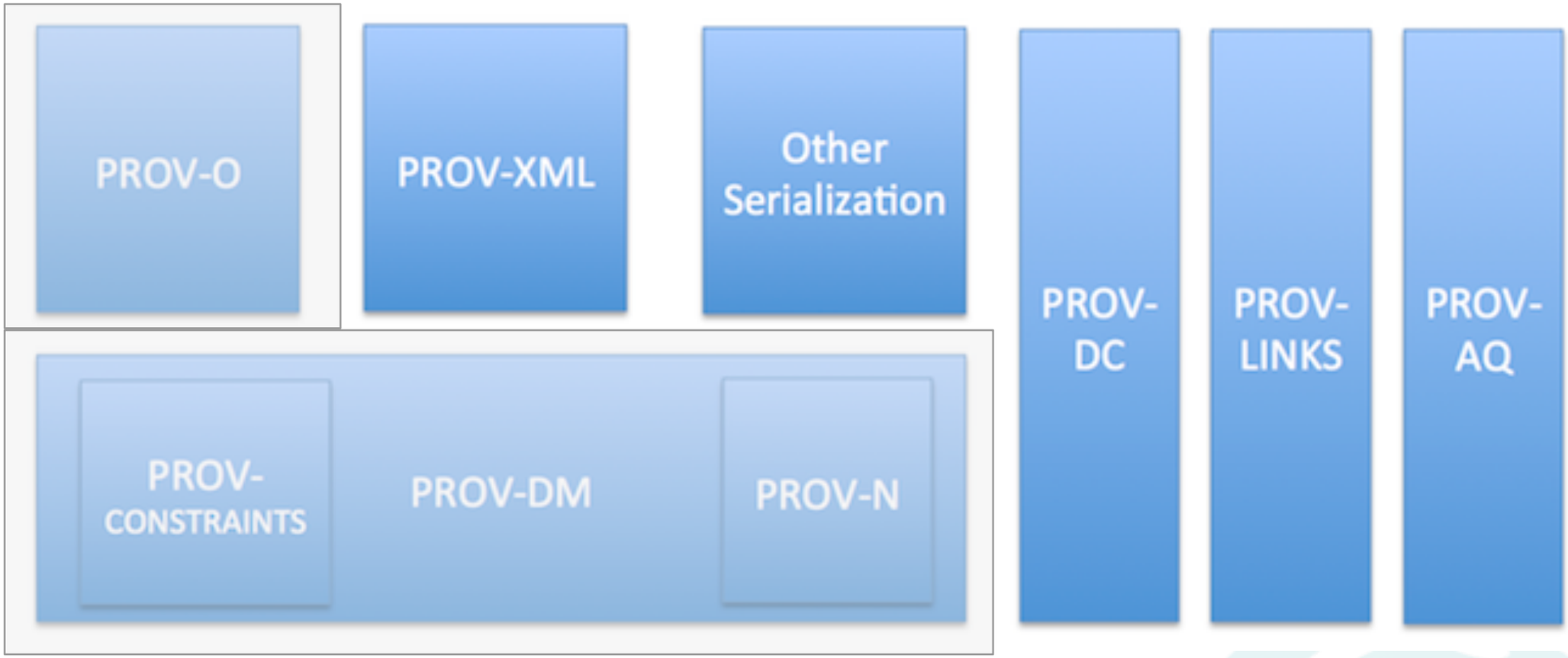
Provenance of the Process:
- account of the evolution of the workflow across versions



FREIRE, J., KOOP, D., SANTOS, E., AND SILVA, C. T. Provenance for Computational Tasks: A Survey. Computing in Science and Engineering 10, 3 (2008), 11–21.

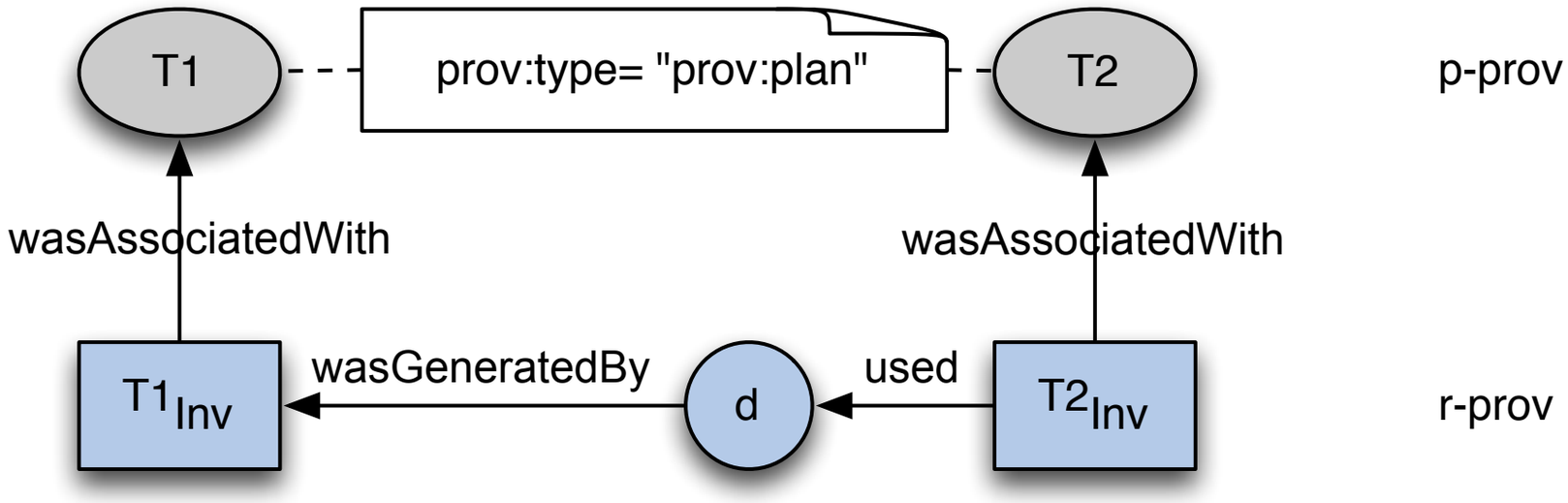
PROV @W3C: scope and structure

Recommendation track



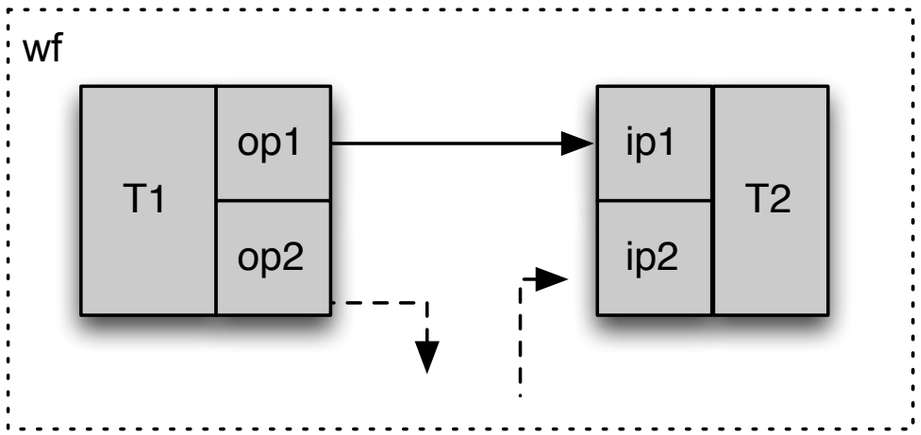
source: http://www.w3.org/TR/prov-overview/

r-prov and p-prov in plain PROV



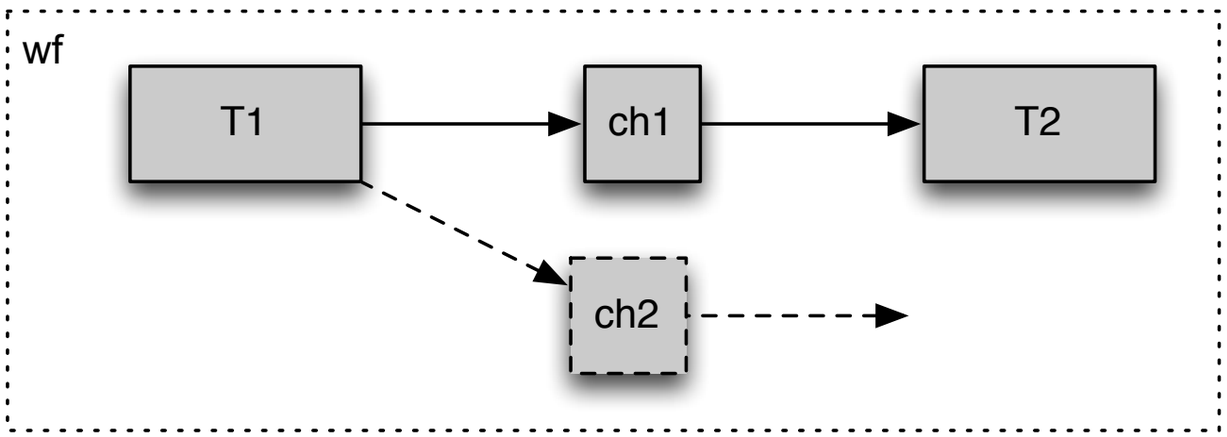

```
// p-prov: tasks, but no data or activities
entity(T1, [ prov:type = 'prov:plan'])
entity(T2, [ prov:type = 'prov:plan'])
// r-prov - task invocation and data
activity(T1inv)
activity(T2inv)
entity(d)  // data flowing between two task instances
wasGeneratedBy(d, T1inv)
used(T2inv, d)
// connecting r-prov and p-prov
wasAssociatedWith(T1inv, _, T1)  // T1 is the plan for T1inv
wasAssociatedWith(T2inv, _, T2)  // T1 is the plan for T2inv
```

Reference dataflow models



Processors, ports, data links

Kepler
Taverna
VisTrails
e-Science Central
...

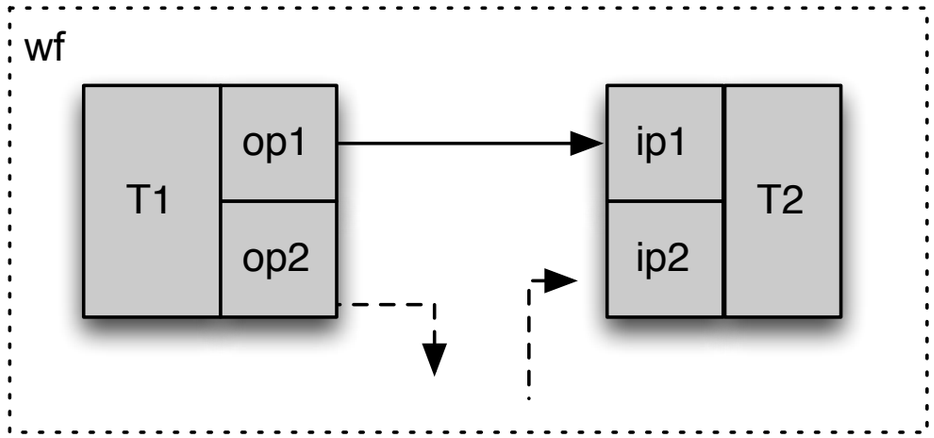


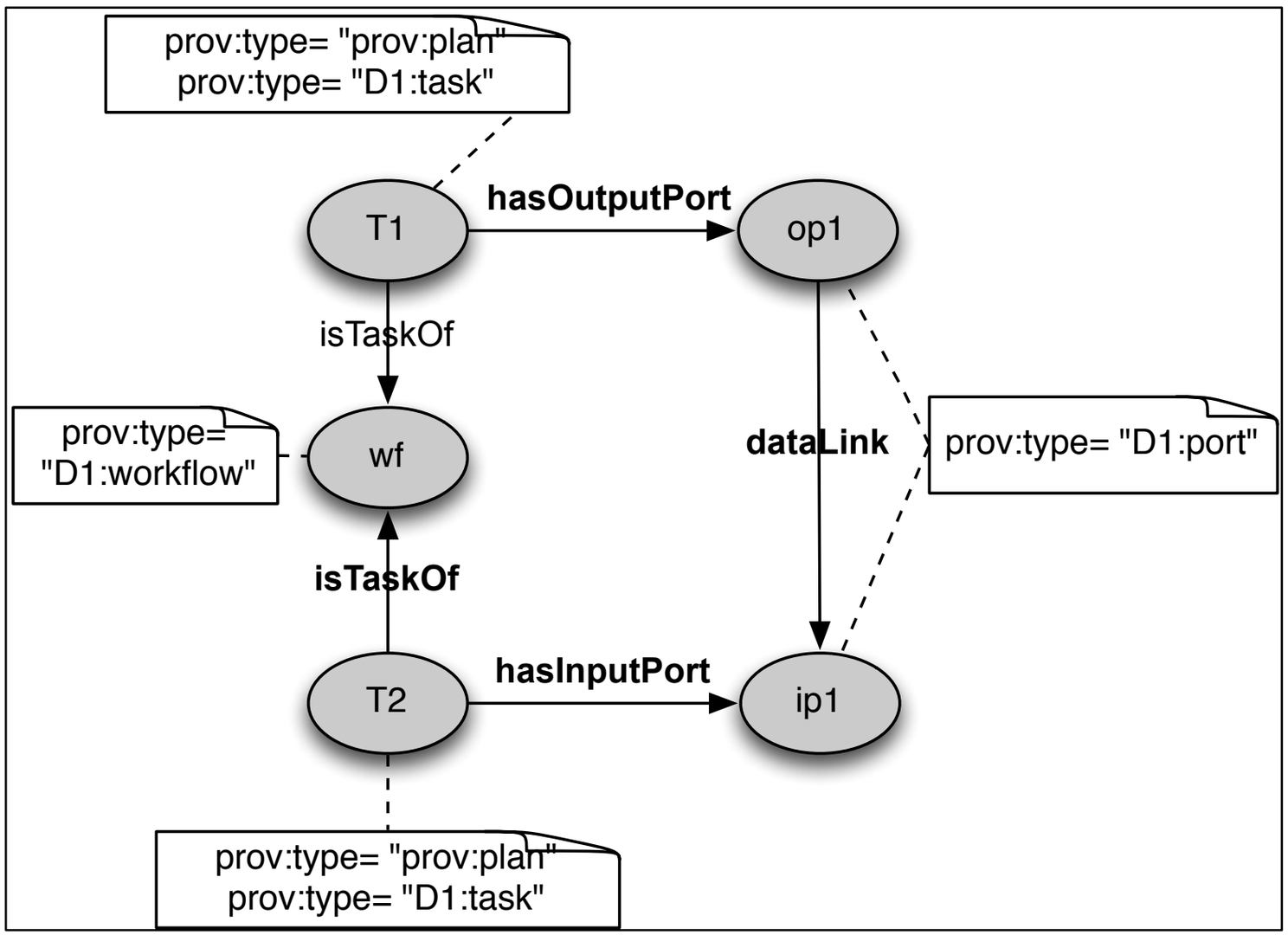
Processors, channels

Dataflow process networks (*)
(e.g. a specific Kepler semantics)

(*) LEE, E. A., AND PARKS, T. M. Dataflow process networks. Proceedings of the IEEE 83, 5 (1995), 773–801.

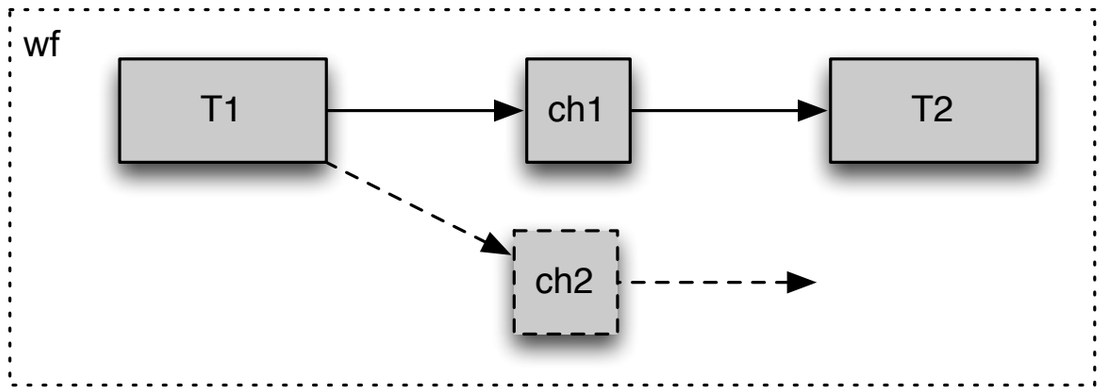
Extensions I / p-prov / ports model

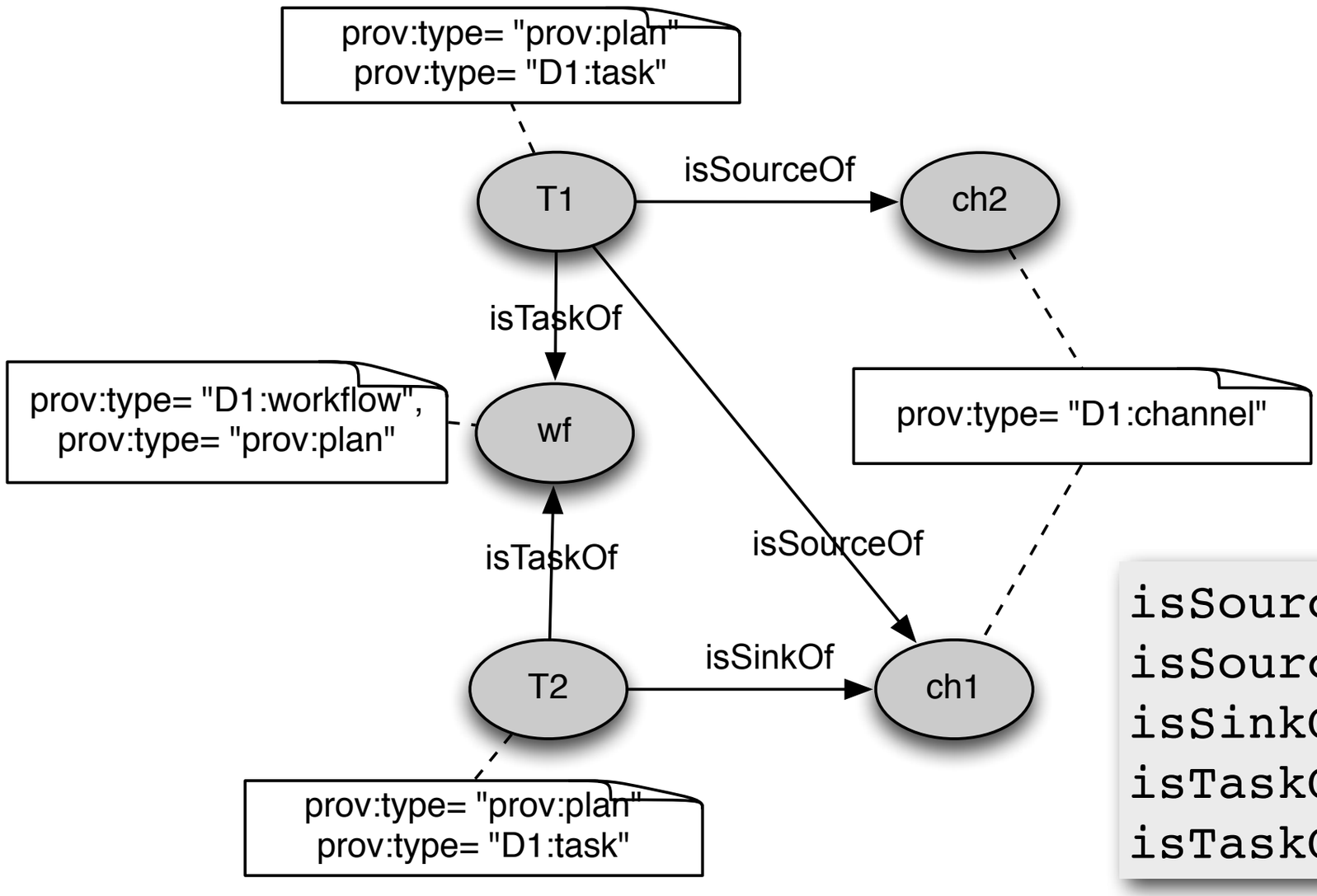





```
hasOutPort(T1, op1)
hasInPort(T2, ip1)
dataLink(op1, ip1)
isTaskOf(wf, T1)
isTaskOf(wf, T2)
```

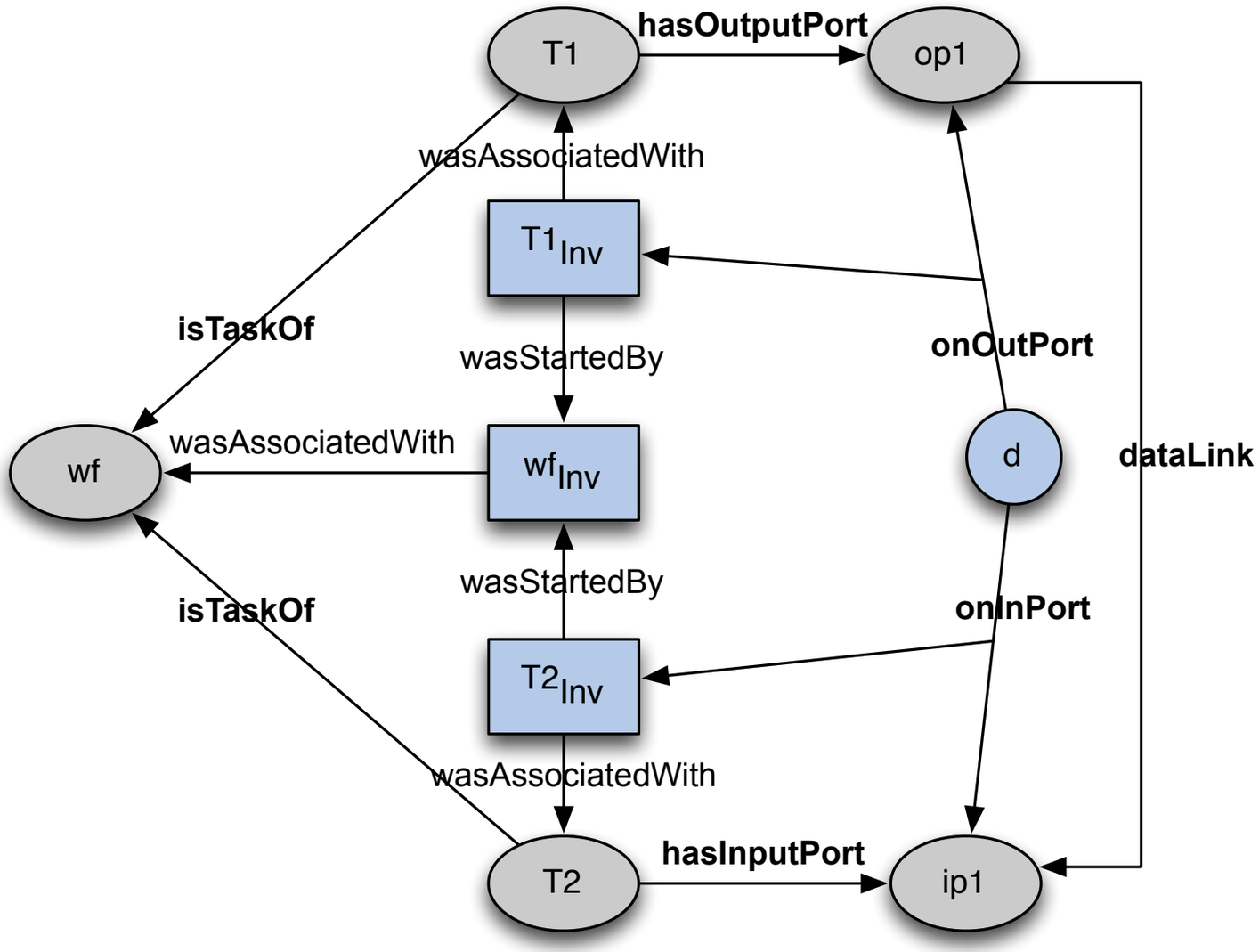
Extensions II / p-prov / channel model






```
isSourceOf(T1,ch2)
isSourceOf(T1,ch1)
isSinkOf(T2,ch1)
isTaskOf(T1, wf)
isTaskOf(T2, wf)
```

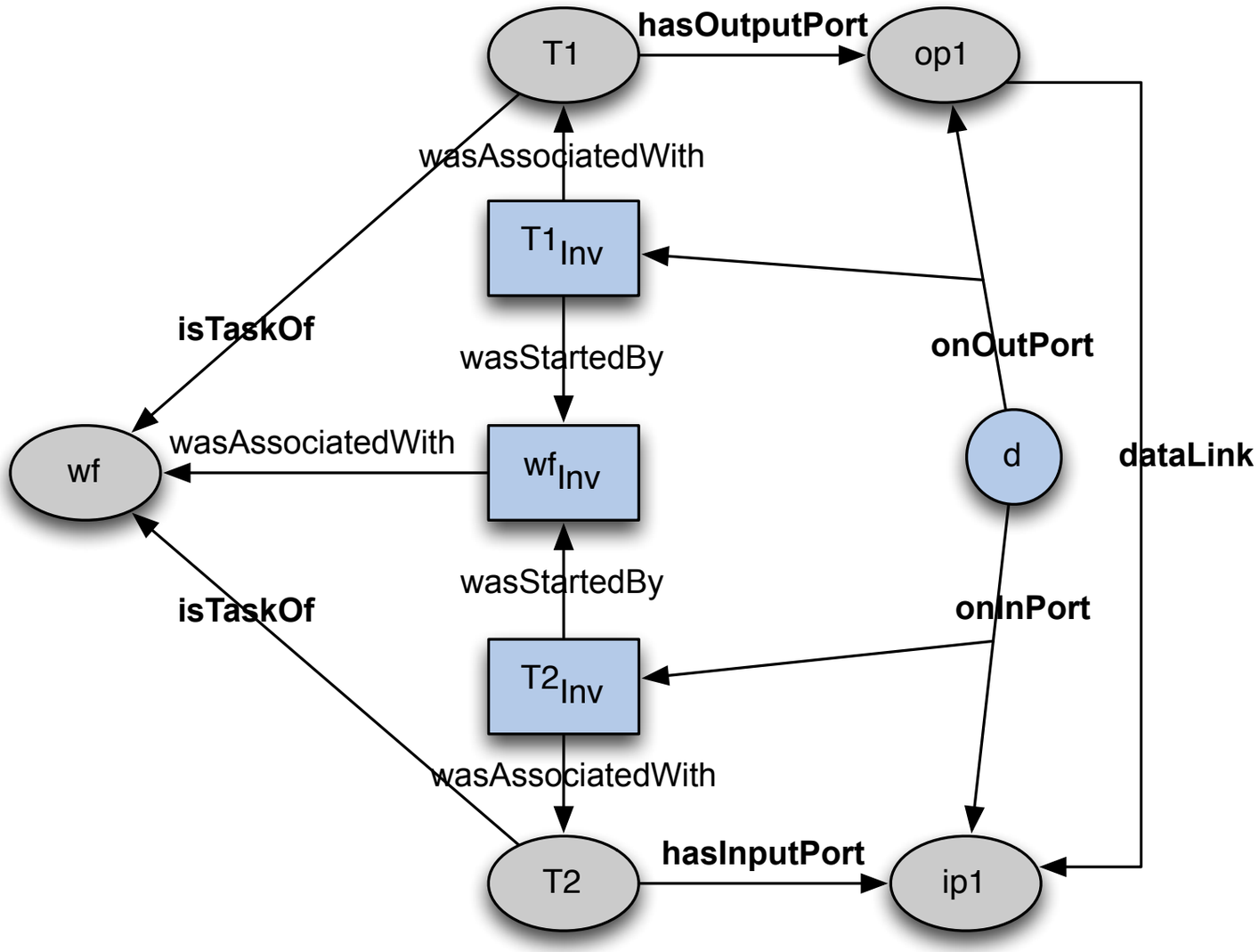
p-prov/r-prov pattern for port-oriented workflows




```
hasOutPort(t1, op1)
hasInPort(t2, ip1)
dataLink(op1, ip1)
isTaskOf(wf, t1)
isTaskOf(wf, t2)

activity (wfRun)
activity ( t1inv )
activity ( t2inv )
entity (d)
onOutPort(d, op1, t1Inv)
onInPort(d, ip1, t2Inv)
```

p-prov/r-prov pattern for port-oriented workflows



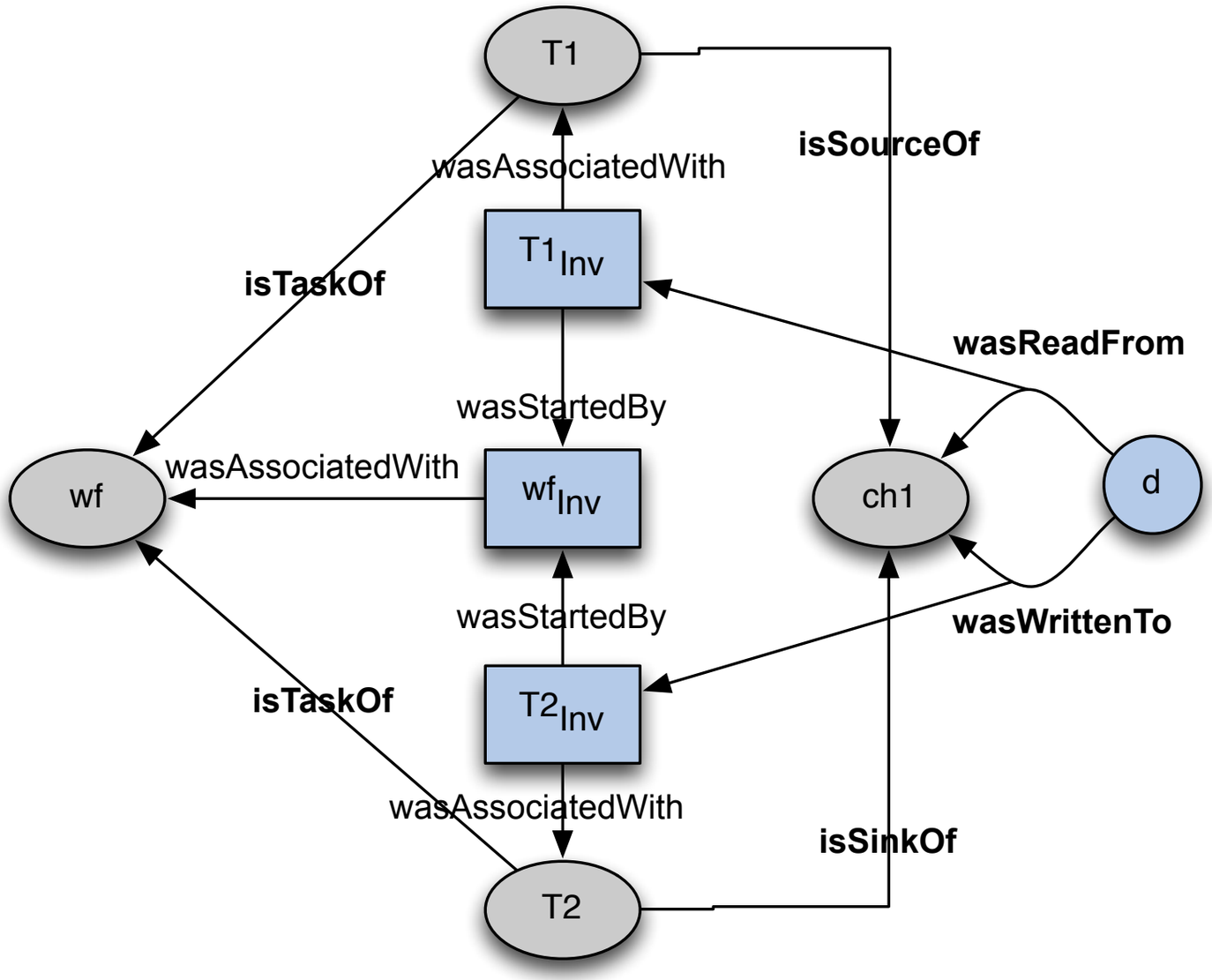

```
hasOutPort(t1, op1)
hasInPort(t2, ip1)
dataLink(op1, ip1)
isTaskOf(wf, t1)
isTaskOf(wf, t2)

activity (wfRun)
activity ( t1inv )
activity ( t2inv )
entity (d)
onOutPort(d, op1, t1Inv)
onInPort(d, ip1, t2Inv)
```

Lossy mapping to plain PROV: Port removal

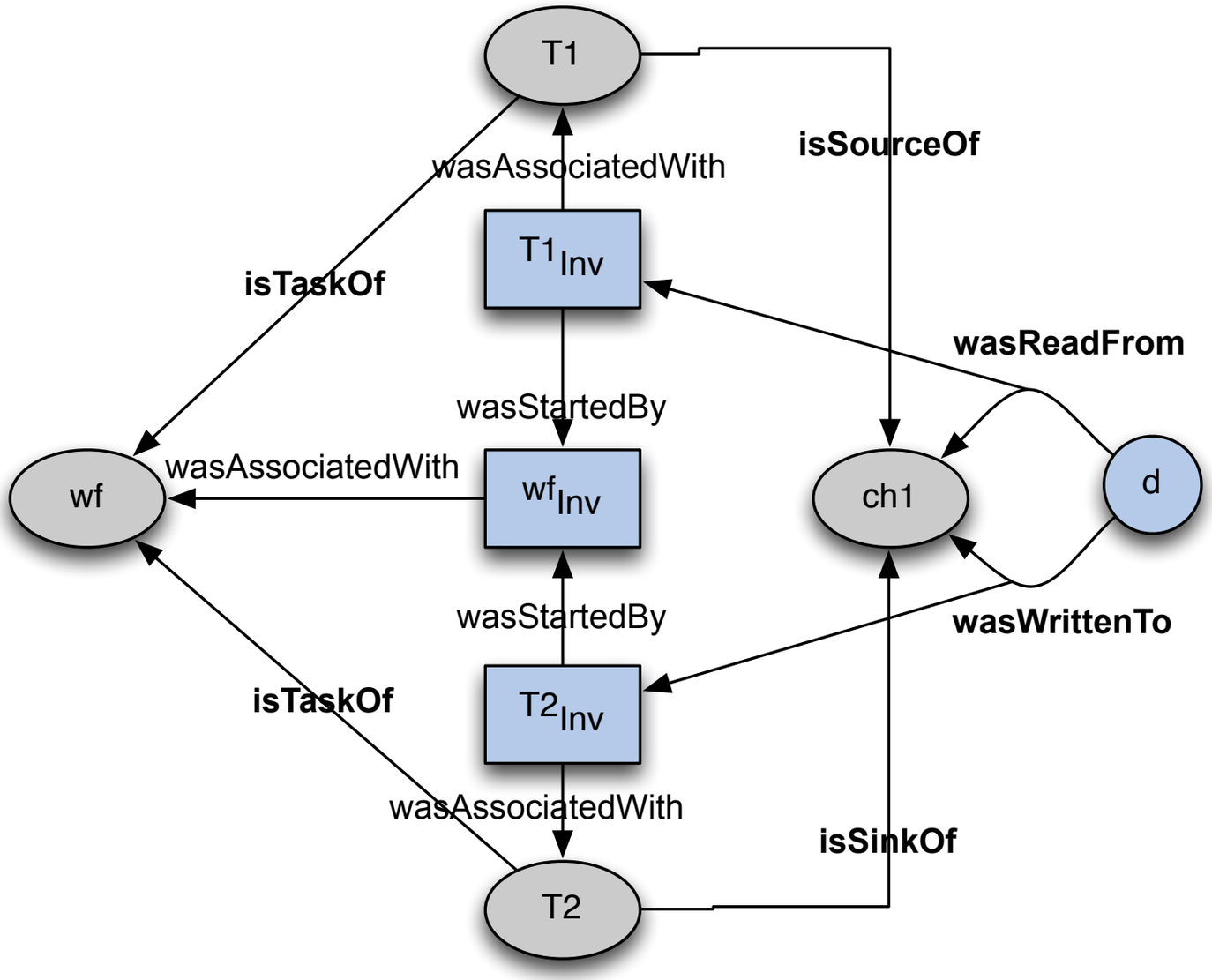
```
wasGeneratedBy(D, tInv) :- onOutPort(D, _, tInv ).
used(tInv, D)           :- onInPort(D, _, tInv)
```

p-prov/r-prov pattern for channel-oriented workflows




```
sourceOf(t1,ch)
sinkOf(t2,ch)
isTaskOf(t1, wf)
isTaskOf(t2, wf)
activity (wfRun)
activity ( t1inv )
activity ( t2inv )
entity (d)
wasWrittenTo(d,ch, t1Inv)
wasReadFrom(d,ch, t2Inv)
```

p-prov/r-prov pattern for channel-oriented workflows



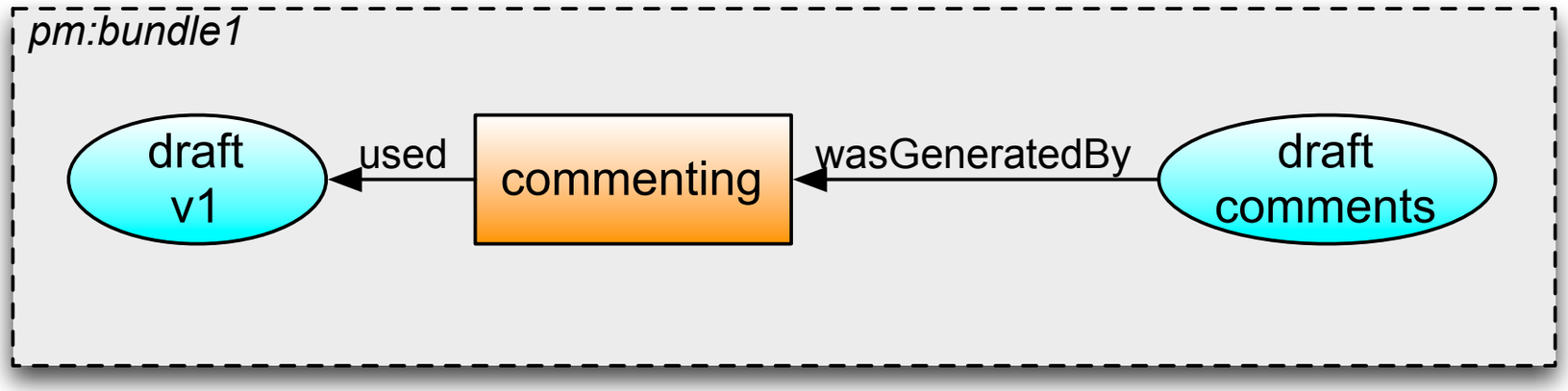

```
sourceOf(t1,ch)
sinkOf(t2,ch)
isTaskOf(t1, wf)
isTaskOf(t2, wf)
activity (wfRun)
activity ( t1inv )
activity ( t2inv )
entity (d)
wasWrittenTo(d,ch, t1Inv)
wasReadFrom(d,ch, t2Inv)
```

Lossy mapping to plain PROV: Channel removal:

```
wasGeneratedBy(d, tInv) :- wasWrittenTo(d,ch, t1Inv )
used(tInv, D):-wasReadFrom(d,ch,t2Inv)
```

Bundles, provenance of provenance

*A **bundle** is a named set of provenance descriptions, and is itself an entity, so allowing provenance of provenance to be expressed.*




```
bundle pm:bundle1

entity(ex:draftComments)
entity(ex:draftV1)

activity(ex:commenting)
wasGeneratedBy(ex:draftComments, ex:commenting,-)
used(ex:commenting, ex:draftV1, -)
endBundle
...
entity(pm:bundle1, [ prov:type='prov:Bundle' ])
wasGeneratedBy(pm:bundle1, -, 2013-03-20T10:30:00)
wasAttributedTo(pm:bundle1, ex:Bob)
```

Structural workflow nesting using bundles

Repurposing: use bundles to associate a workflow execution with the provenance it generates

```
entity (wfRunTrace, [ prov:type='prov:Bundle' ])
wasGeneratedBy(wfRunTrace,wfInv,-)
```

This makes it possible to write hierarchical provenance of nested workflows, recursively:

```
entity (T, [prov:type="D1:task", prov:type="D1:workflow"])
```

```
bundle wfRunTrace
  activity(wfRun)  // run of top level wf
  activity(Tinv)   // run of T, a sub-workflow

  wasAssociatedWith(Tinv, _, T)
  entity(TinvTrace, [ prov:type='prov:Bundle' ])
  wasGeneratedBy(TinvTrace, Tinv, _)
  ...
endbundle
```

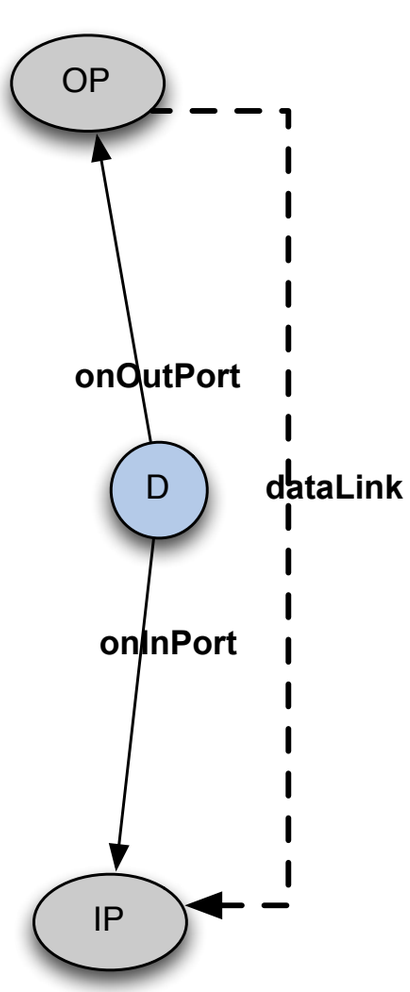
Answering the sample queries

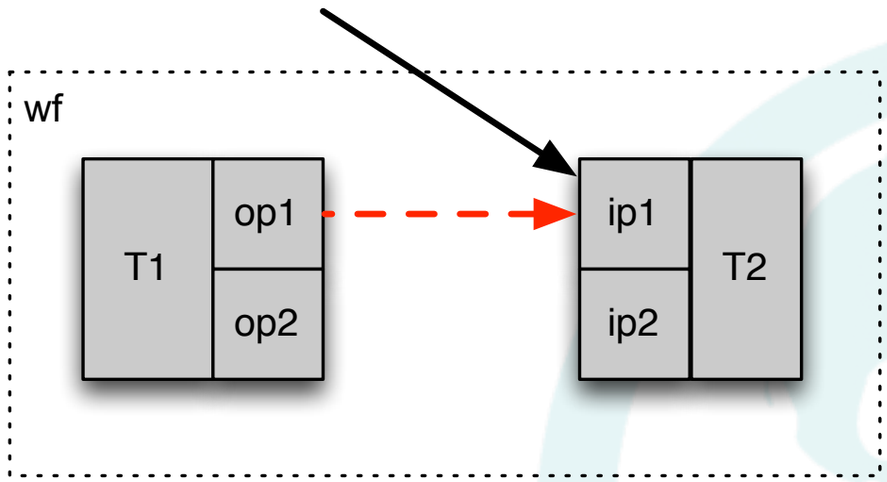
Q3: “match the provenance trace to the workflow structure”

Two steps:

- define rules that entail p-prov relations from r-prov relations, and
- check that those new p-prov relations are consistent with any constraints defined on the workflow structure / infer new p-prov statements

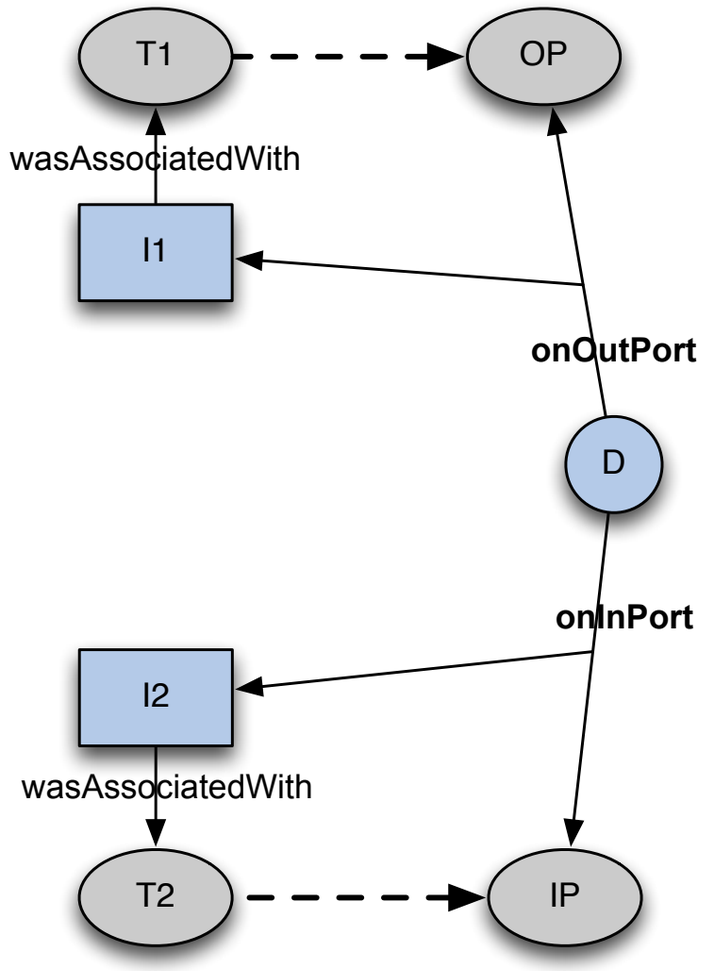
```
dataLink (OP, IP) :- onOutPort(D,OP,_),
                     onInPort(D,IP,_).
```





constraint violation?

Structure inferences

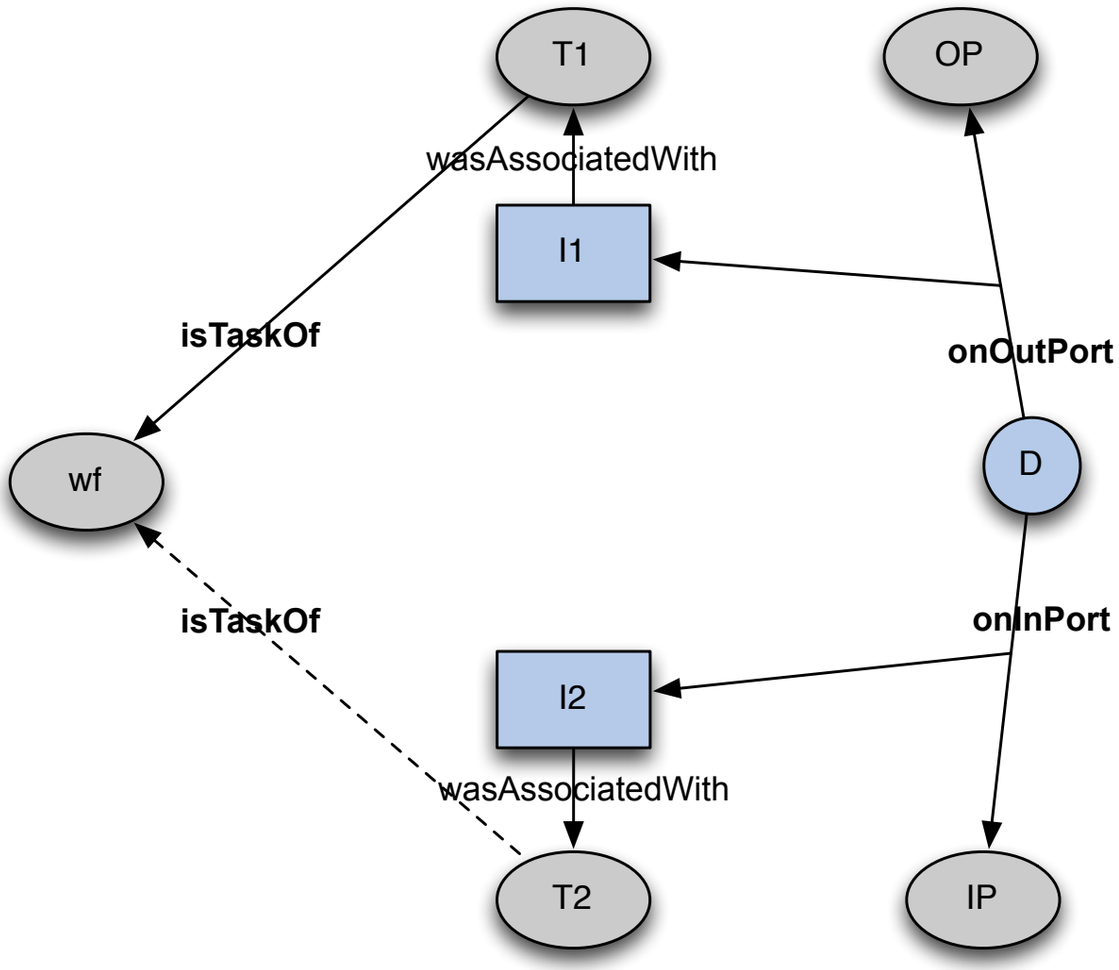



```
hasOutPort(T, OP) :-
        onOutPort(D,OP,I1),
        wasAssociatedWith(I1,_,T1).
```

```
hasInPort(T, IP) :-
        onInPort(D,IP,I1),
        wasAssociatedWith(I1,_,T1).
```

Structure inferences

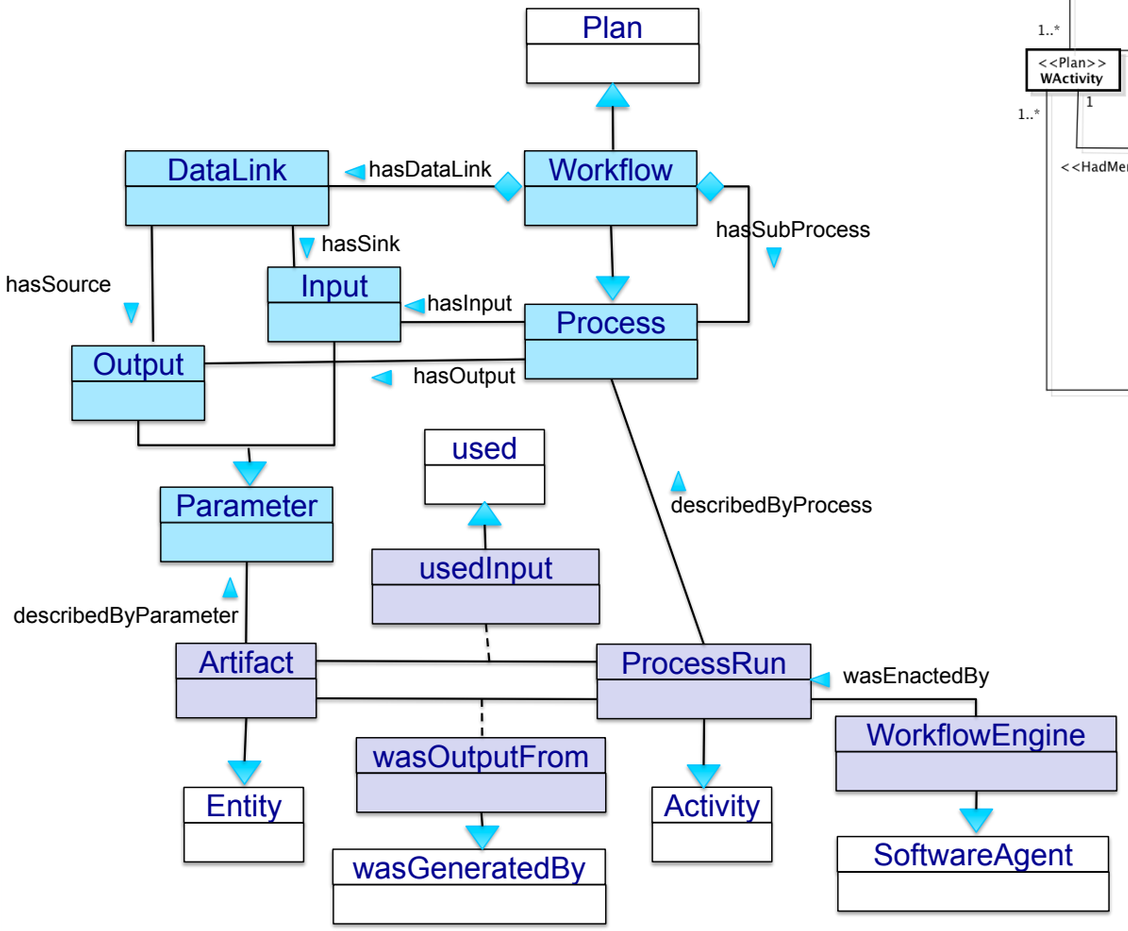
```
isTaskOf(T2, Wf) :-     onOutPort(D,OP,I1),
                        hasOutPort(T1, OP),
                        onInPort(D,IP,I2),
                        hasInPort(T2, IP),
                        wasAssociatedWith(I1,_,T1),
                        wasAssociatedWith(I2,_,T2),
                        isTaskOf(T1, Wf).
```



Other PROV extensions into “workflow-land”

Prov-Wf:

Flavio Costa, Vítor Silva, Daniel de Oliveira, Kary Ocaña, Eduardo Ogasawara, Jonas Dias, and Marta Mattoso, *Capturing and Querying Workflow Runtime Provenance with PROV: a Practical Approach*, Procs. BigProv’13, Genova, Italy, March 2013



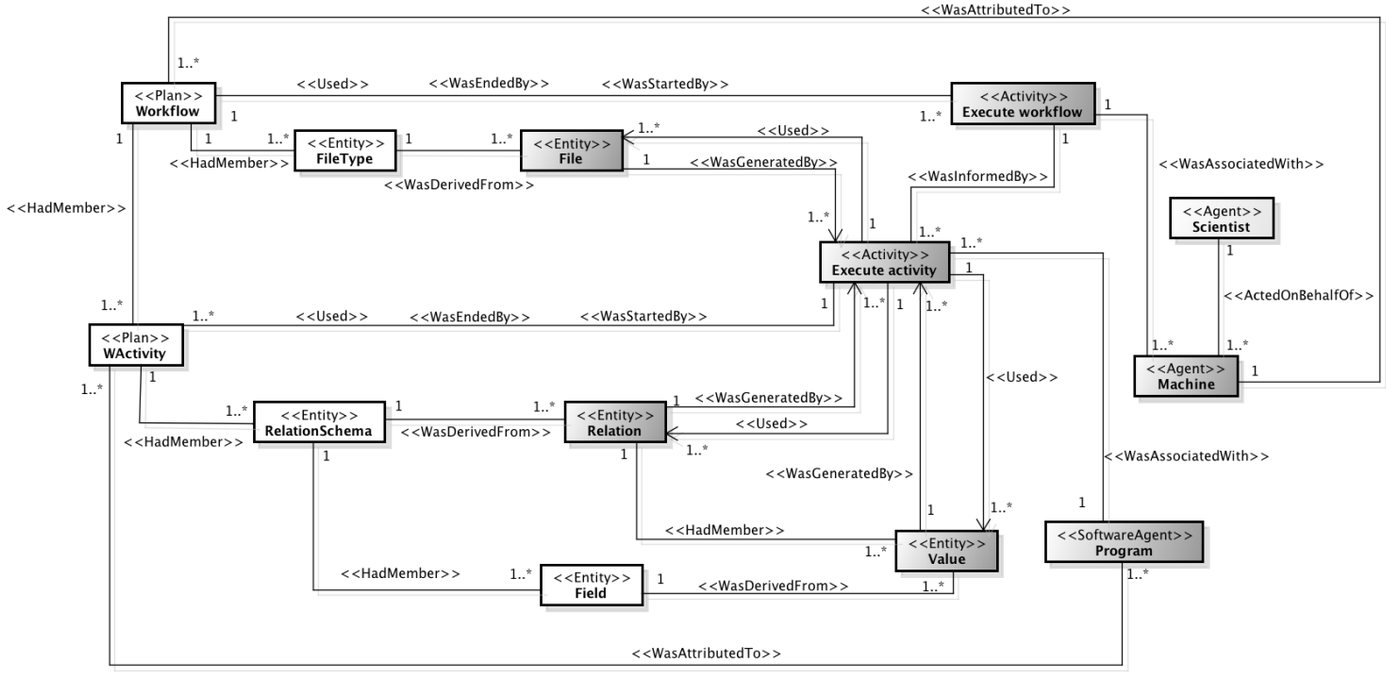


Figure 1 PROV-Wf data model

WfProv from the Wf4Ever project

www.wf4ever-project.org

Summary and extensions

Entity types:	D1:workflow, D1:port, D1:task, D1:channel
p-prov Relations:	
taskOf(t, wf)	task t is part of workflow wf
hasOutPort(t,p)	task t has output port p
hasInPort(t,p)	task t has input port p
dataLink(p1, p2)	a data link connects port p1 to p2
sourceOf(t,c)	task t is the source of channel c
sinkOf(t,c)	task t is the sink of channel c
r-prov Relations:	
onInPort(d, p, tInv)	data d was observed on input port p
onOutPort(d, p, tInv)	data d was observed on output port p
wasWrittenTo(d,c,tInv)	data entity d was written to channel c
wasReadFrom(d,c,tInv)	data entity d was read from channel c

- Simple extensions to PROV
  - designed to model p-prov
  - complementary to r-prov
- They enable queries that cut across r-prov and p-prov
- Bundle mechanism used for provenance of nested workflow components
- Next step: harmonize similar extensions proposed by other groups
  - overall goal is to achieve interoperability