

PSOA Prova: PSOA Translation of Pure Production Rules to the Prova Engine

Lukas Grätz

joined work with: Harold Boley Adrian Paschke

Institut für Informatik & Institut für Philosophie, Universität Leipzig, Germany
lukas[DT]graetz[AT]studserv.uni-leipzig.de

Workshop – Rules: Logic and Applications
19-20 December 2018, Athens

originally presented during RuleML+RR 2018, Luxembourg

Table of Contents

- 1 Introduction
- 2 Use Case: Royal Family
- 3 PSOA RuleML for Reconstruction
 - Extension: Runtime KB Consult and Unconsult
- 4 Demonstration
 - Succession by a Derivation Rule
 - Succession by a Pure Production Rule
- 5 Results

Use Case: Royal Family



- How did Prince William become a successor?
- How to represent succession with KR?

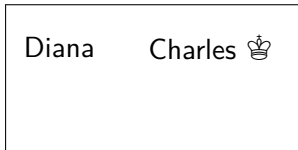
Use Case: Royal Family



- How did Prince William become a successor?
 👉 *Succession to the Crown Act*
- How to represent successorship with KR?
 👉 *Our challenge*

Royal Family

1.



2.



3.



4.

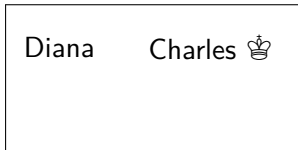


5.

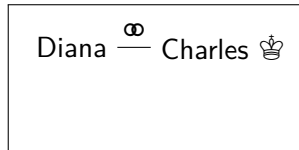


Royal Family

1.



2.



3.



4.

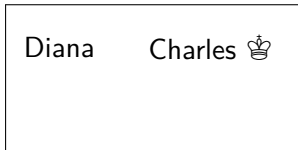


5.

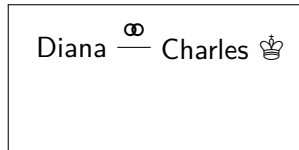


Royal Family

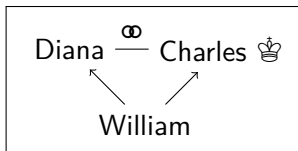
1.



2.



3.



4.

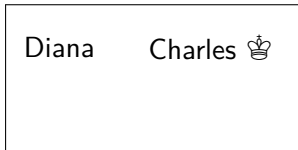


5.

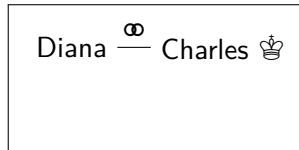


Royal Family

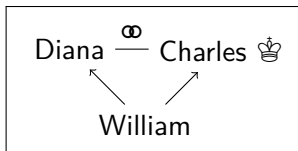
1.



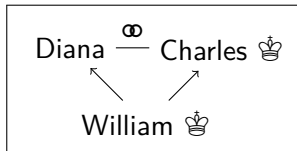
2.



3.



4.

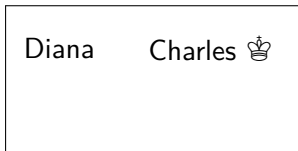


5.

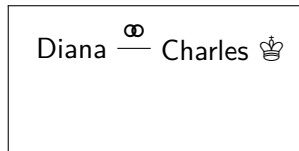


Royal Family

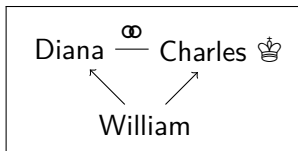
1.



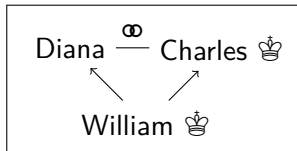
2.



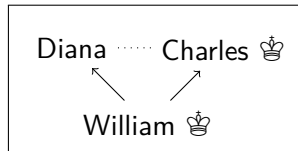
3.



4.



5.



- object-relational knowledge

- `Object#Predicate`
- `Object#Predicate(key+>value)`
- `Object#Top (key->value)`
- `Object#Predicate(elem_1 elem_2 ...)`
- `Object#Top (-[elem_1 elem_2 ...])`
- `OID#Pred(d1... -[e1...] k1+>v1 k2->v2 ...)`

membership
dependent slot
independent slot
dependent tuple
independent tuple
all together

- derivation rules

```
forall ?Q1 ?Q2 ... ?Qn (  
  <conclusion> :- And( <cond_1> <cond_2> ... )  
)
```

- object-relational knowledge

- `Object#Predicate`
 - `Predicate(key+>value)`
- `Object#Top` (`key->value`)
- `Predicate(elem_1 elem_2 ...)`
- `Object#Top` (`-[elem_1 elem_2 ...]`)
- `OID#Pred(d1... -[e1...] k1+>v1 k2->v2 ...)`

membership
dependent slot
independent slot
dependent tuple
independent tuple
all together

- derivation rules

```
forall ?Q1 ?Q2 ... ?Qn (  
  <conclusion> :- And( <cond_1> <cond_2> ... )  
)
```

Extension: Runtime KB Consult and Unconsult

New feature in PSOA Prova: `consult` and `unconsult` at run-time

```
> consult RoyalFamily-KB2.psoa
```

Extension: Runtime KB Consult and Unconsult

New feature in PSOA Prova: `consult` and `unconsult` at run-time

```
> consult RoyalFamily-KB2.psoa
```

Translated KB:

```
prdsloterm('_1', '_marriage', '_partner', '_Diana').  
prdsloterm('_1', '_marriage', '_partner', '_Charles').  
memterm('_1', '_marriage').
```

Extension: Runtime KB Consult and Unconsult

New feature in PSOA Prova: `consult` and `unconsult` at run-time

```
> consult RoyalFamily-KB2.psoa
```

Translated KB:

```
prdsloterm('_1', '_marriage', '_partner', '_Diana').  
prdsloterm('_1', '_marriage', '_partner', '_Charles').  
memterm('_1', '_marriage').
```

```
> unconsult RoyalFamily-KB2.psoa
```

Extension: Runtime KB Consult and Unconsult

New feature in PSOA Prova: `consult` and `unconsult` at run-time

```
> consult RoyalFamily-KB2.psoa
```

Translated KB:

```
prdsloterm('_1', '_marriage', '_partner', '_Diana').  
prdsloterm('_1', '_marriage', '_partner', '_Charles').  
memterm('_1', '_marriage').
```

```
> unconsult RoyalFamily-KB2.psoa
```

```
> marriage( partner+>Diana partner+>Charles )
```

Translated Query:

```
prdsloterm(Q1, '_marriage', '_partner', '_Diana'),  
prdsloterm(Q1, '_marriage', '_partner', '_Charles'),  
memterm(Q1, '_marriage').
```

Answer(s):

No

Succession by a Derivation Rule

```
forall ?Ch ?P1 ?P2 (  
  ?Ch#successor :-  
    And( ?Ch#child( parent->?P1 parent->?P2 )  
         marriage( partner+>?P1 partner+>?P2 )  
         ?P1#successor  
    )  
)
```

Succession by a Derivation Rule

```
forall ?Ch ?P1 ?P2 (  
  ?Ch#successor :-  
    And( ?Ch#child( parent->?P1 parent->?P2 )  
         marriage( partner+>?P1 partner+>?P2 )  
         ?P1#successor  
    )  
)
```

↪ Translation into Prolog:

```
memterm(QCh, '_successor')  
:- sloterm(QCh, '_parent', QP1),  
   sloterm(QCh, '_parent', QP2),  
   memterm(QCh, '_child'),  
   prdsloterm(Q1, '_marriage', '_partner', QP1),  
   prdsloterm(Q1, '_marriage', '_partner', QP2),  
   memterm(Q1, '_marriage'),
```

Succession by a Pure Production Rule

Definition

A *pure production rule* is an extended derivation rule, where the derived conclusion is asserted persistently to the KB. If the condition holds, the conclusion is derivable; moreover, the conclusion will be asserted at least before the condition becomes unsatisfied.

Notation: `<conclusion> ::- <condition>`

Succession by a Pure Production Rule

Definition

A *pure production rule* is an extended derivation rule, where the derived conclusion is asserted persistently to the KB. If the condition holds, the conclusion is derivable; moreover, the conclusion will be asserted at least before the condition becomes unsatisfied.

Notation: `<conclusion> ::- <condition>`

```
forall ?Ch ?P1 ?P2 (
  ?Ch#successor ::-
    And( ?Ch#child( parent->?P1 parent->?P2 )
         marriage( partner+>?P1 partner+>?P2 )
         ?P1#successor
    )
)
```



Demonstration

([screenrecord](#) – just in case)

Evaluation of a Pure Production Rule

Events to start evaluation $\hat{=}$ `assert` the conclusion:

① Structure operation

- ① after `consult`
- ② before `unconsult`
- ③ ...

② Behavior invocation

- ① conclusion is derivation (sub)goal
- ② ...

③ Clock (e.g. polling)

④ External

⑤ ...

transparent for the user!

- ✓ PSOATransRun fork on [Github](#)
- ✓ PSOATransRun[PSOA2Prova,Prova]
- ✓ Consult and unconsult
- ✓ Pure production rules

Wiki

The End