Mark Proctor

Project Lead

redhat | JBoss a division of Red Hat

Mark Proctor

Project Lead

Drools

Business Logic integration Platform /

Guvnor

Expert

Fusion

Flow

The SkyNet funding bill is passed.
The system goes online on August 4th, 1997.
Human decisions are removed from strategic defense.
SkyNet begins to learn at a geometric rate.
It becomes self-aware at 2:14am Eastern time, August 29th
In a panic, they try to pull the plug.
And, Skynet fights back

Drools | BLiP

Drools Expert

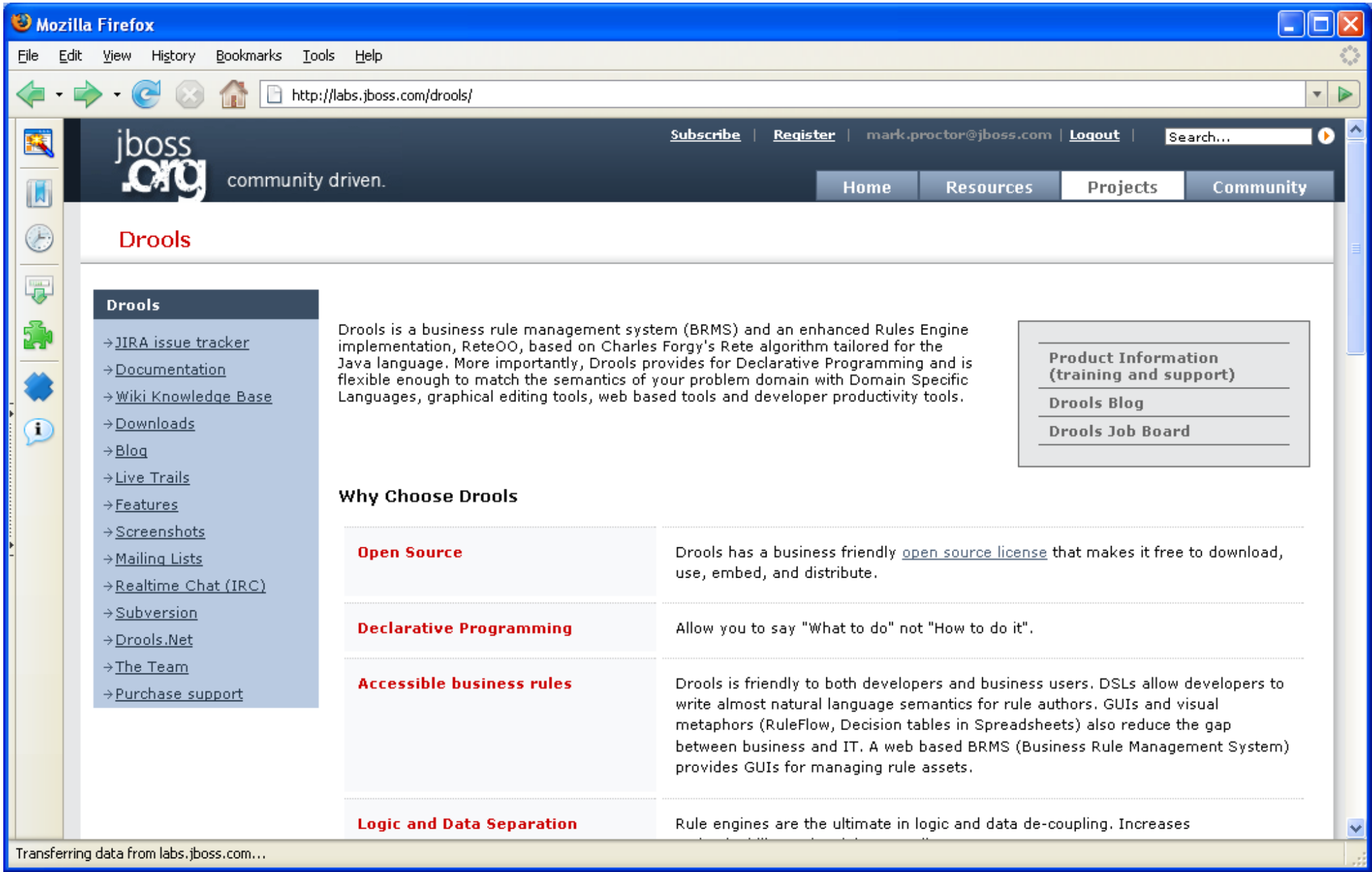
Drools Flow

Drools Fusion

Drools Guvnor

Business Logic integration Platform

Drools | http://www.jboss.org/drools



Drools | http://blog.athico.com

Drools - Business Logic integration Platform - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://blog.athico.com/

Most Visited home Java Google KDE GNOME PostgreSQL GROKLAW Yell.com Radio Times Dictionary Nopaste TinyURL! jira The System R-DEVICE

Drools Boot Camp T-Shirts

Posted by Mark Proctor

T-Shirts Finally Arrived today, so everyone was very excited :) Thought I'd put up some photo's of our motley crew - Asif and Andrea (who have been here the other days) could not make it today, so they missed out on the photo. Just click any of the photos to enlarge.

- DotNet (1)
- DRL (1)
- Drools (138)
- Drools Boot Camp (2)
- Drools Flow (5)
- drools puzzle (4)
- droos ide update-site downloads (1)
- DSL regexp antlr (1)
- DynaBeans (1)
- dynamically generated classes (1)
- Eclipse (4)
- ESP (2)
- examination (2)
- expressiveness (3)
- FactTemplate (1)
- Forward Chaining (1)
- generated classes (1)
- GIS (2)
- grammar (1)
- GSoC (3)
- GUI (6)
- Guice (1)
- Guvnor (9)
- IKVM (1)
- image processing (1)
- interview (1)
- Janino (1)
- java (2)
- JavaOne (1)
- javapolis (2)
- JBoss Rules (53)
- jBPM (2)
- JDT (1)
- Jess (2)
- JFDI (1)
- Job (4)
- JUG (2)
- KAMS (1)
- LDAP (1)
- machine learning (3)
- MicroContainer (1)
- Mind Map (1)
- MISMO (2)
- modify block (1)
- Monitoring (1)
- MVEL (6)
- MySQL (1)

Done

Drools | Syntax improvements

Direct fields supports by most systems

- Person(name == “mark”)

Eval workarounds for nested accessors and method calls

- Person(eval(address.city == “London”))
- Person(eval(pets[0].someMethod() == 30))

Return Value workarounds for nested accessors and method calls

- Person(age == ($otherPerson.age + 15))
  Person(age == ($otherPerson.someMethod()))

Drools | Syntax improvements

Nested accessors currently supported

- Person(address.city == “London”)
- Person(pets[0].name == “rover”)
- Person(pets[rover].age > 30)

Method calls not supported

- Person(pets[0].someMethod() == 30)

Unbracketed expressions not support

- Person(age == $otherPerson.age + 15)

Drools | Syntax improvements

```
$n : Number( intValue > 100 )
        from accumulate(  $s : StockTicker( symbol = “RHAT”  )
                                                over window:time( 5s ),
                                    average( $s.price ) )
```

```
$n : accumulate(  $s : StockTicker( symbol = “RHAT”  )
                                                over window:time( 5s ),
                            average( $s.price ) > 100 )
```

Else

Else allowed, but only works on last evaluate as end CE:

```
when
    $Person()
    evaluate( $person.getPets().get(0).age > 20 )
then
    ....
else
   Print( “pet is old” );
end
```

Labelled else evals, can be used on any eval:

```
when
    $Person()
    [else1] eval( $person.getPets().get(0).age > 20 )
then
    ....
else [else1]
   Print( “pet is old” );
end
```

Why can't I put else on Patterns themselves and fields, to avoid eva

```
when
    [else1]  $Person( pets[0].age > 30 )
then
    ....
else [else1]
   Print( “pet is old” );
end
```

Otherwise

Sometimes you just want a catch all situation,
“else” does not provide this.

If I have 3 rules, each one checks a different capital location:
“london”, “paris” or“new york”
Yet I insert a fact where the capital is “athens”, how do I handle this Unknown?

This is typically handled by decision tables by the use of the “otherwise” value, which actually generates the following rule:
capital not in (“london”, “paris, “new york”)

Drools | Otherwise

Ignoring performance issues when this list gets very large, there is more the issue that this is a solution for tooling – it's not a good solution for direct drl authoring and requires manual maintenance.

Enter "otherwise-group". We can put all the related rules that we would like to apply this scenario too into a group.

We then add a rule that will handle that "otherwise" situation:
Capital == OTHERWISE
The engine recognises that this is a special value and that the rule is part of a "otherwise-group". What will happen is that for a given propagation on that ObjectType is none of the other fields match then the OTHERWISE is considered to match.

This allows a fine grained and sophisticated way to handling unknown values.

Logical Closures

```
when
    $l : Light( status == “on” )
then
     … do some stuff....

     localClosure( $l ) {
         println( “light has gone off”
     }
end
```

Logical Modify

```
when
    Document( status == “valid” )
    $p : Person()
then
     logicalModify( $p ) {
         status = “valid”
     }
end
```

What happens if multiple rules “logical modify” the same field and one loses it's justifications?

Duration, Repetition and Cron

```
rule “name”
    duration 1m30s
when
    $l : Light( status == “on” )
then
   SendEmail( “turn the light off” )
```

```
rule “name”
    duration 1m30 repeat
when
    $l : Light( status == “on” )
then
   SendEmail( “turn the light off” )
```

Duration, Repetitation and Cron

Field Name	Mandatory?	Allowed Values	Allowed Special Characters
Seconds	YES	0-59	, - * /
Minutes	YES	0-59	, - * /
Hours	YES	0-23	, - * /
Day of month	YES	1-31	, - * ? / L W
Month	YES	1-12 or JAN-DEC	, - * /
Day of week	YES	1-7 or SUN-SAT	, - * ? / L #
Year	NO	empty, 1970-2099	, - * /

```
rule “name”
   cron (  0 0/15 * * * * )
when
    $l : Light( status == “on” )
then
   sendEmail( “turn the light off” )
```

Execution Groups

Agenda Groups

```
rule
    agenda-group “A”
when
    ...
```

RuleFlow Groups

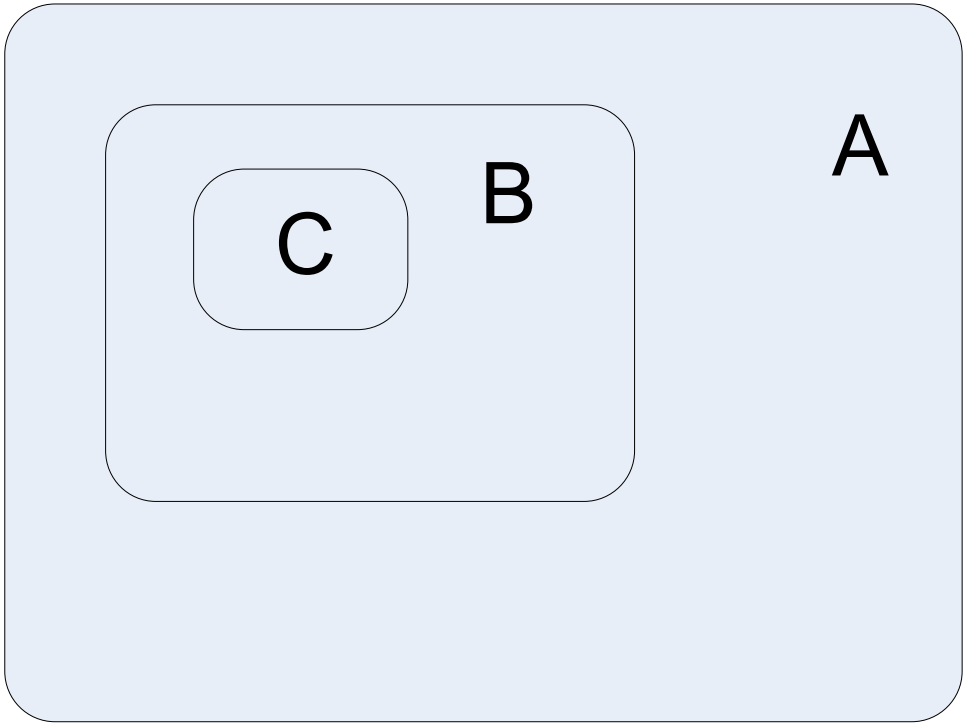
```
rule
    ruleflow-group “A”
when
    ...
```

Activation Groups

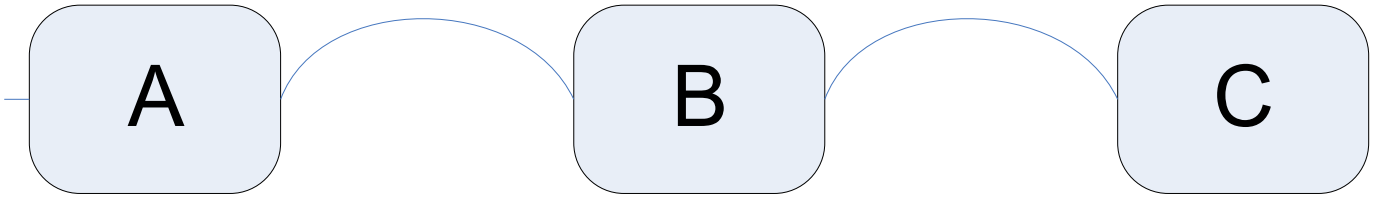
```
rule
    activation-group “A”
when
    ...
```

Drools | Execution Groups

Agenda Group
•Push/Pop stack



Rule Flow



Drools | Execution Groups

Agenda Groups
•setFocus() to push
•single global stack

Rule Flow Groups
•Can evaluate groups in “parallel”, i.e. multiple process instances.
•Can only be scheduled for evaluation as part of a starting a process instance.
•setFocus() still go to global stack

Activation Group
•Very specialist

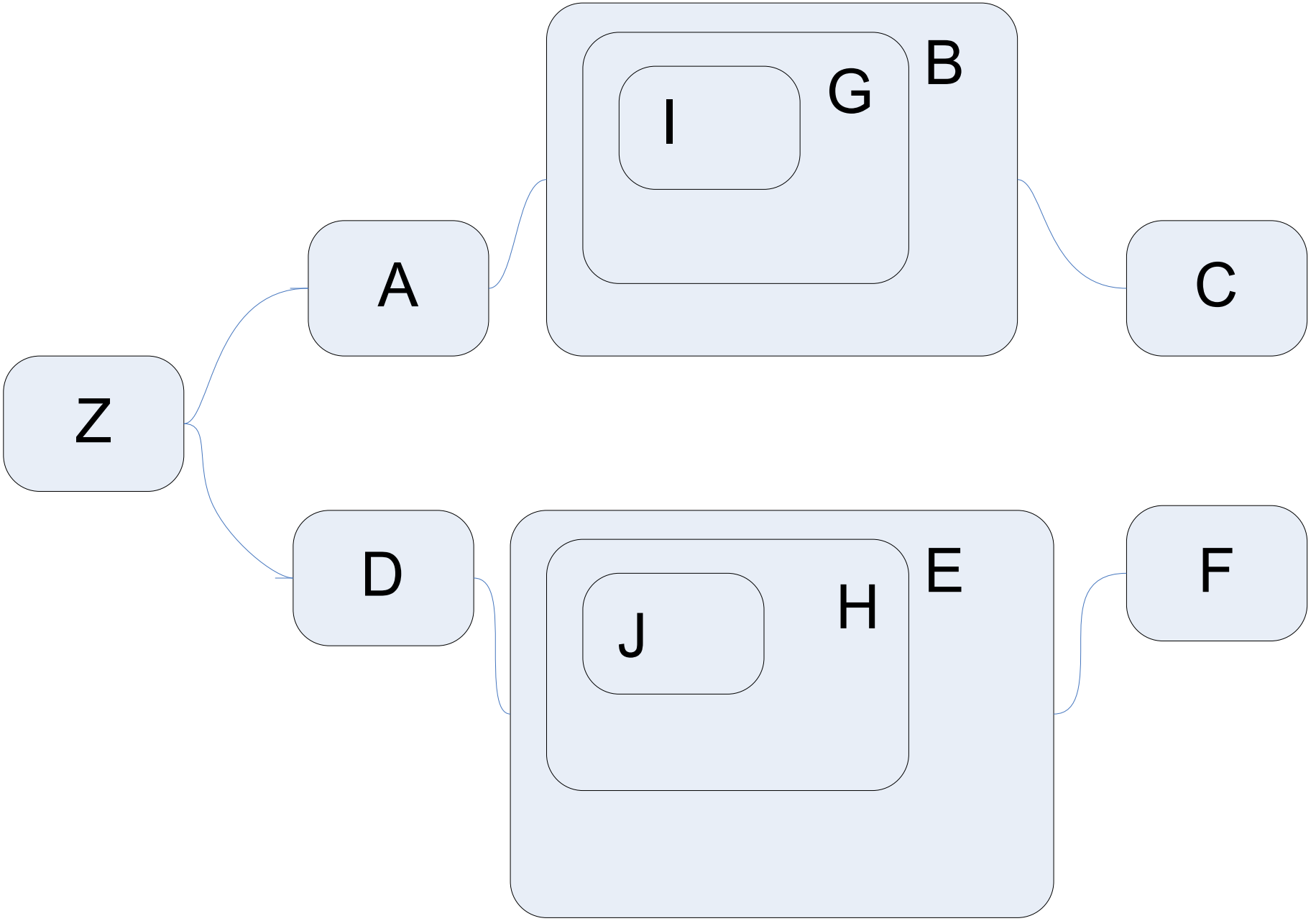
Execution Groups

Generic “execution-group”

- Deprecate ruleflow-group, agenda-group, activation-group keywords
- Any group can be “called” (activations onto the parent agenda)
- Any group can have it's “focus” set, scope goes up one and activations onto it's own agenda.
- “focus” are re-entrant, thus equivalet to push/pop, but always local to parent container
- Any group can be made part of a process (ruleflow-group), effectively the process is just executing “call”.

Drools | Execution Groups

"focus" scoped to parent container



Execution Groups

CRS:

```
execution-group “A”
    crs “depth”
end
```

Execution modes:

```
execution-group “A”
    execution-mode “sequential”
…
```

Activation Group:

```
rule “only fire one”
when
    …
then
    executionGroup.cancel();
end
```

Execution Groups

Meta-Rules

- Control which rules can fire
- Control rule firing order
- Guards, which can block further group (or rule) evaluation, until true

MVCC and Transactions

MVCC – Multi Version Concurrency Control

- Facts are time or counter stamped
- Modify causes a “clone” of the fact which will be used for that transaction.
- Other transactions must filter out in-transaction facts that have been modified for “isolation”.
- Execution-groups themselves can be transaction boundaries.
- Or we can select regions of groups to form a transaction boundary

MVCC and Transactions

MVCC – Multi Version Concurrency Control

- Allows concurrent reads with the write.
- Concurrent write will result in one transaction either being blocked or failing.
- Transaction boundary analysis can provide some clues to optimistic or pessimistic locking – i.e. is one transaction region potentially impacted by side effects from the other transaction.
- What happens if the action is from outside engine and that fact is is already part of a transaction? Deep/shallow?

Positional Slotted Language

- Production rule systems like slotted facts
- Prolog likes positional
- POSL provides both worlds

class, position is assumed in field declaration order

```
Person {
    String name;
    String location;
    int age;
}
```

positional

```
Person("darth", "london", 105 );
```

slotted

```
Person( name = "darth", location = "london", age = 105 };
```

mixed positional and slotted instantiation

```
Person( "darth", location = "london", age = 105 }
```

Data:
Person("darth", "london", 105);
Person("yoda", "london", 200);
Person("luke", "paris", 40);

Slotted query :
Person($n, "london", $y);
Person("darth", "london", 105);
Person("yoda", "london", 200);

positional query:
Person($n : name, location == "london", $y : age);

mixed query:
Person($n, location == "london", $y : age);
Person($n, $y : age, location == "london");
Person($n, "london", $y : age);

Existing Drools Queries, more like SQL, all arguments are "in":

```
query queryName1(arg1, arg2, arg3)
    $o1 : Object1( field1 = arg1 );
          Object2( field1 = $o1, field2 = $arg3)
end
```

and querries are called with positional only

```
query( “value1”, “value”, “value3” );
```

When calling lets allow arguments to be specified or have variables passed for unification. We can even then call other queries.

```
query queryName1(q1arg1, q1arg2, q1arg3)
    $o1 : Object1( field1 = q1arg1 );
          queryName2( $o1, q2arg2 == q1arg2, q2arg3 ==  q1arg3 )
end

query queryName2(q2arg1, q2arg2, q2arg3)
    $o1 : Object2( field1 = q2arg1 );
            Object2( field1 = q2arg2, field2 = q2arg3)
End
```

So now we can all this withpositional and slot:

```
queryName1( “value1”, $a2 : q1arg2, $a1 : q1arg3 )
queryName1( “value1”, $a2, $a1 : q1arg3 )
```

Federated Queries

A query is a name plus the arguments.

If we treat this as an interface, we don't care what returns the results As long as it obeys the POSL invocation.

A query call could call from a Drools predefined query.

Or you could register a prolog data provider, or a hibernate one. That registers the name + arguments. When making invoking the query it looks up from the registry.

A query can call a query and a rule can call a query.

Thus we can have data results that are combined from various sources.

Federated Queries

```
Rule “federated data sources”
    $p : Person()
    $c : Car( owner = $p )
          droolsQuery( $p, $c, $f : someField)
          prologQuery( $f, someField == “someValue” )
when
   ...


 query droolsQuery1(q1arg1, q1arg2, someField)
     $o1 : Object1( field1 = q1arg1 );
               hibernateQuery2( $o1, q2arg2, someField )
 end
```

Federated Queries

If Person does not have positional info, map it.

```
query Person(name, location, age)
    Person( name == name,  location == location, age == age )
end
```

Our belief [by "our belief" we mean the "inference-engine's belief"] about a sentence can be:

•false, the sentence is believed to be unconditionally false; this is also called a contradiction
•true, the sentence is believed unconditionally true; this is also called a premise
•assumed-true, the sentence is assumed true [we may change our belief later]; this is also called an enabled assumption
•assumed-false, the sentence is assumed false [we may change our belief later]; this is also called a retracted assumption
•assumed, the sentence is believed by inference from other sentences
•don't-care.

Other topics

Semantics
•Xml sucks, need compact syntax. Manchester Syntax?
•How to have invalid data. Clips stops and asks the user to correct. Can we get more sophisticated
•Pojo representations.
•Update DRL to infer joins, and other sugar for ontologies

Sequencing
•Rete is a fixed network. Can we have the joins drive a “state machine” so we can detect event sequences.

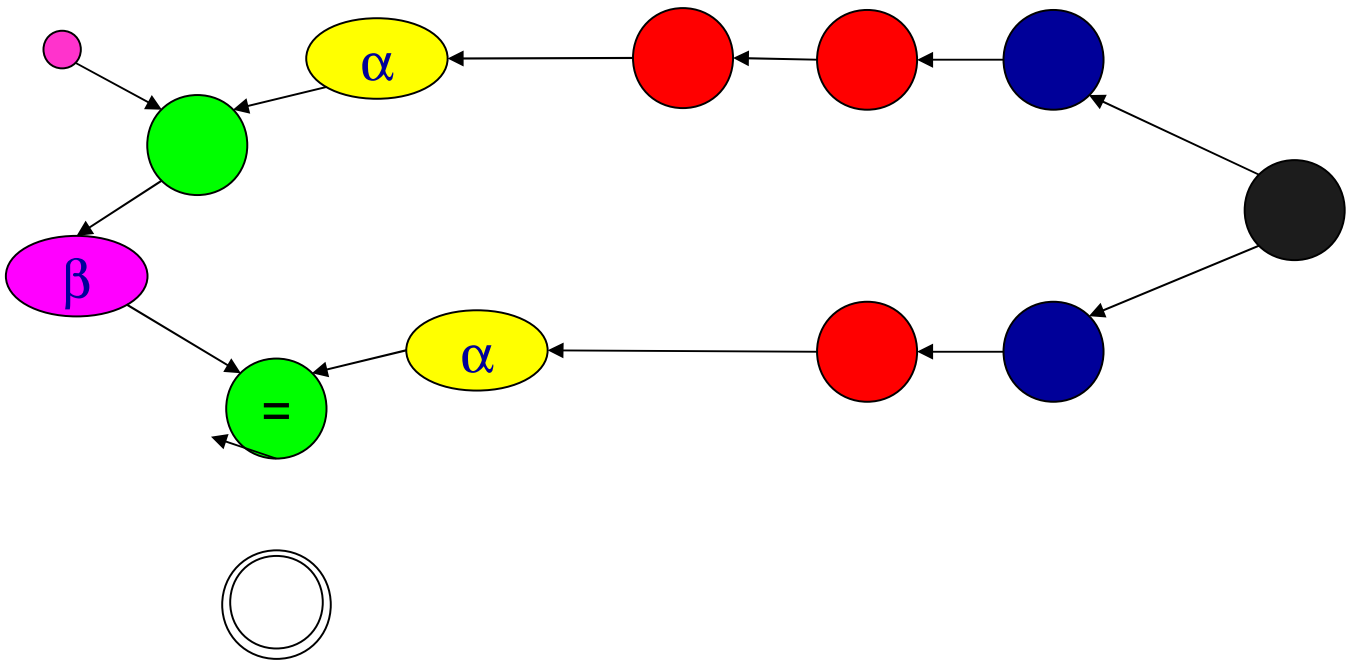
Composition

Auditing
Immutable event based models

Backward chaining/Lazy Fact/Field initialisation

Example

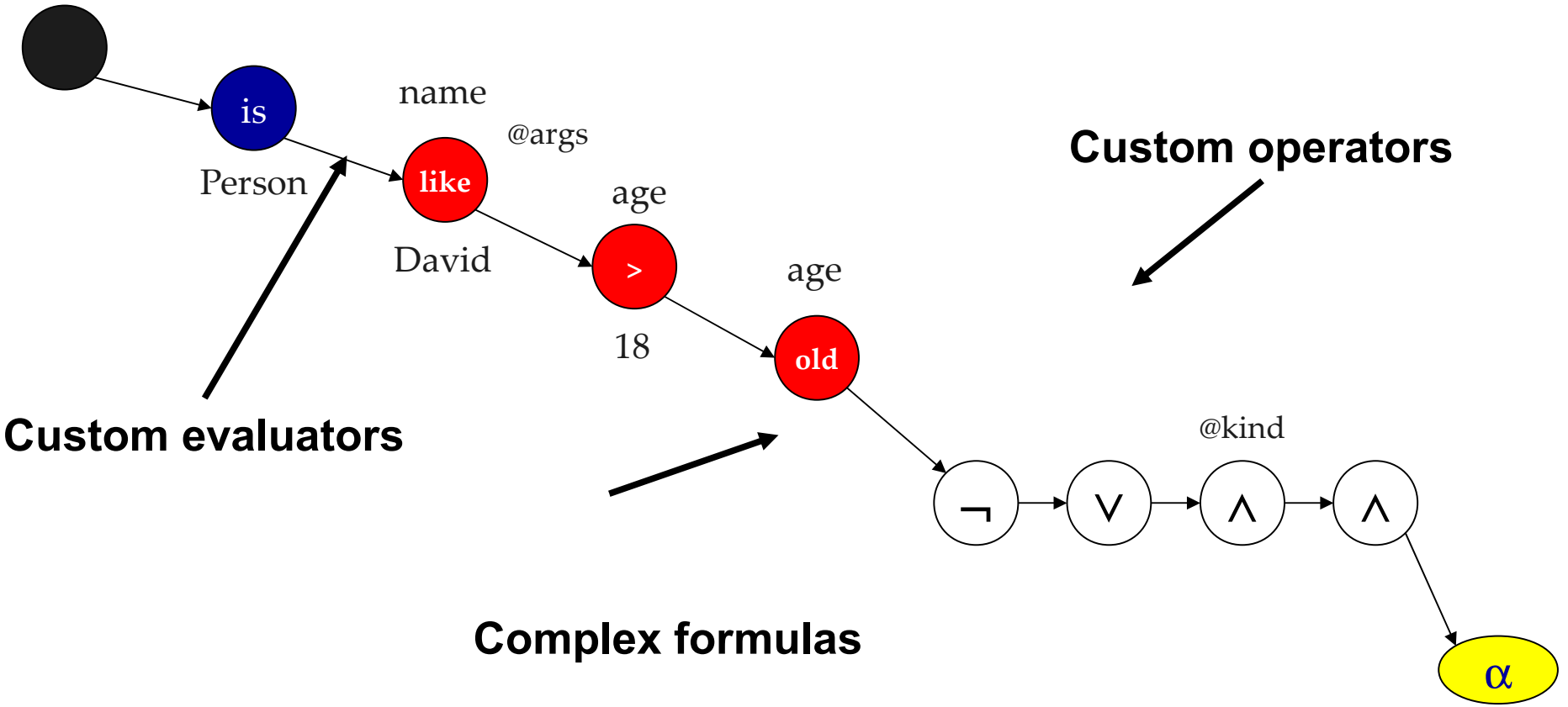
```
Rule "r1"
when
    $p : Person( name == "David" , age > 18)
    $c : Car ( color == red , owner == $p )
then ... end
```



α-network

```
$p : Person( name like @[args="it,en"] "David"
             and @[ kind = "..." ]
             (age > 18 or age neg old) )
```

min or product



β-network

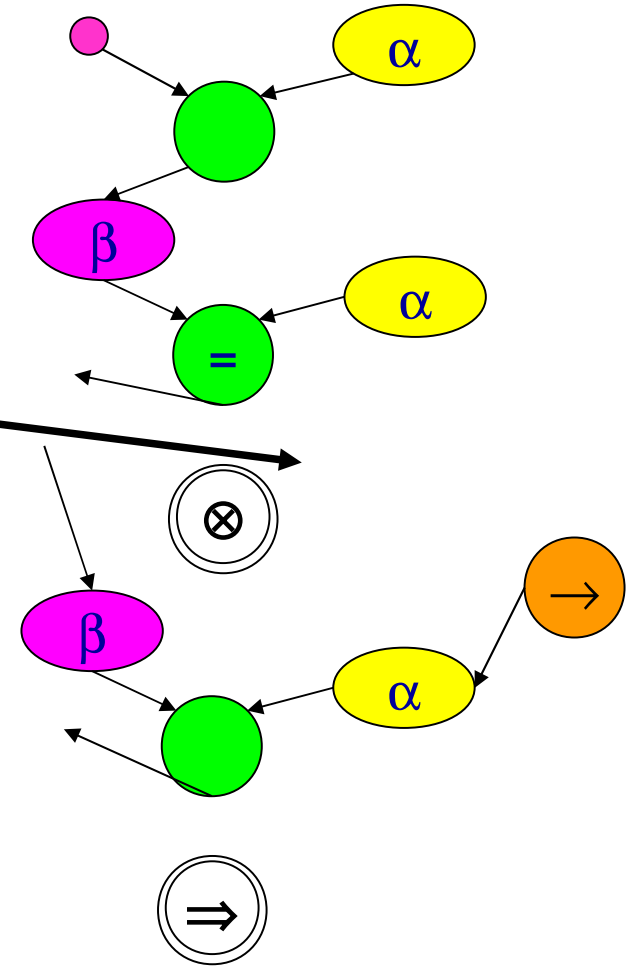
```
Rule "r1"
  degree = "..."
when
  $p : Person(...)
  and
  $c : Car ( owner == @[bool] $p )
then ... end
```

Priors

"Standard" constraints

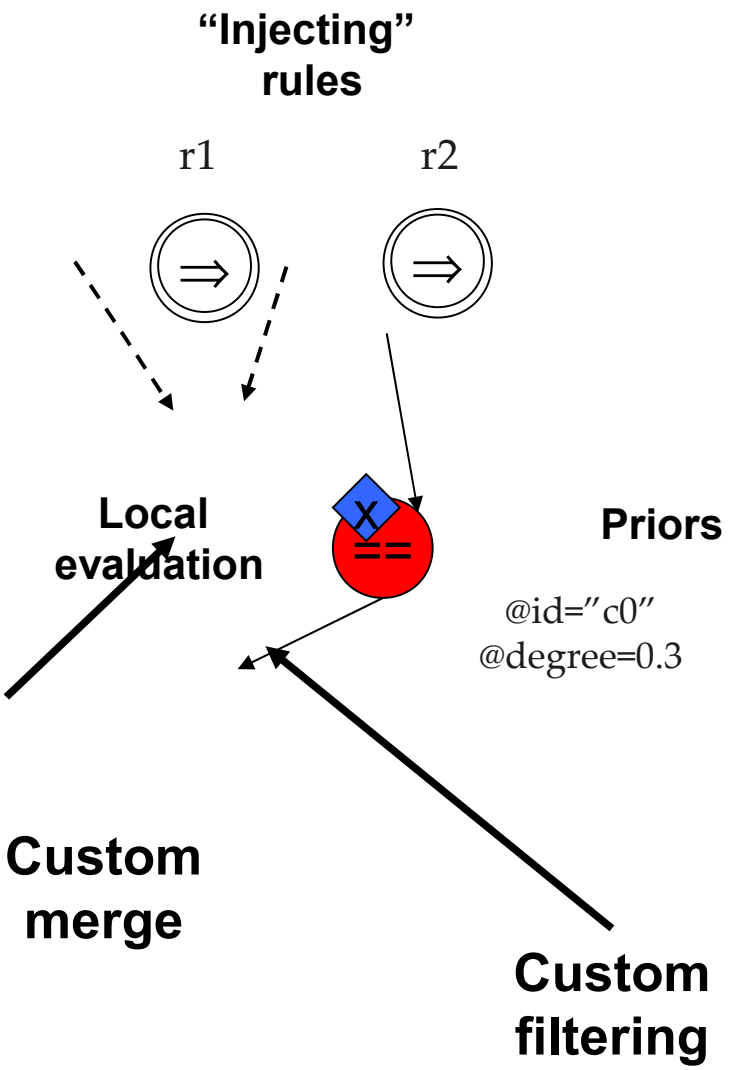
Gradual rules / Probabilistic rules

Custom Deduction



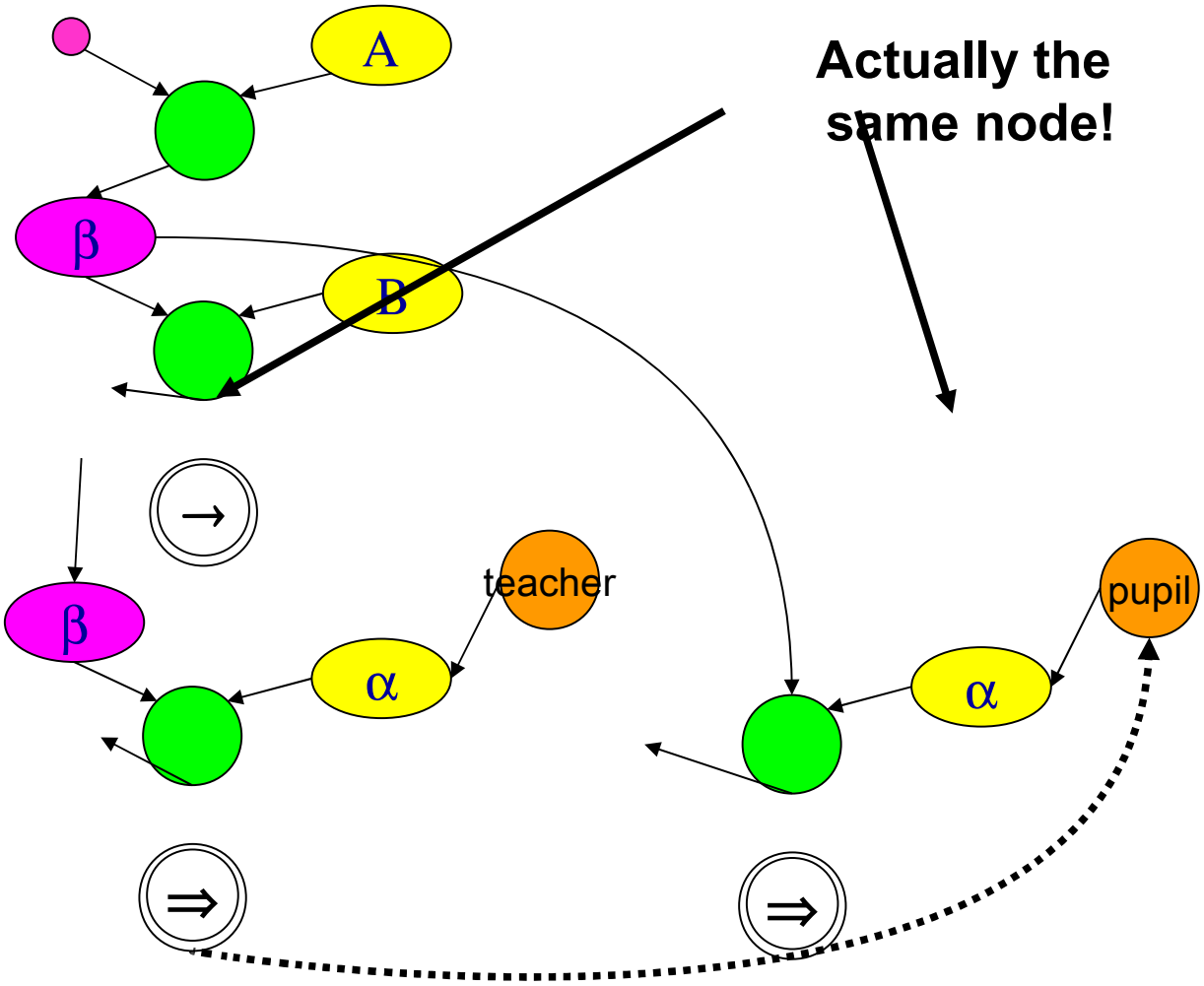
Multiple Evaluation

```
rule "r1"
  ...
then inject(x,"c0") end

rule "r2"
  ...
then inject(x,"c0") end

rule "r"
when
  Type( field == @[ degree="0.3",
                    id="c0",
                    kind="...",
                    params="..." ]
        value)
end
```



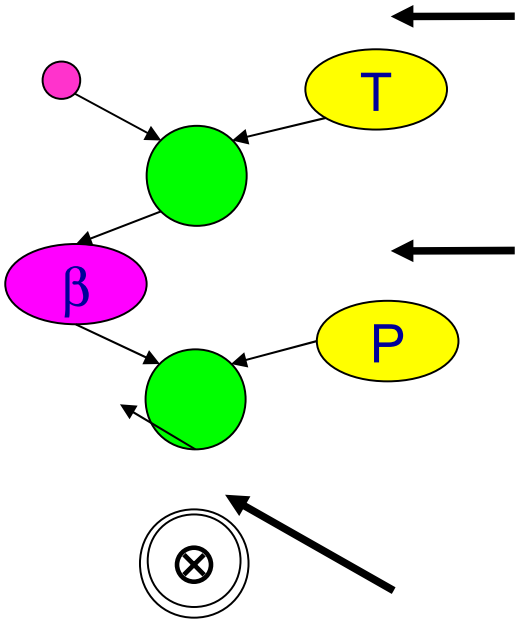
Beyond MP : Induction

```
rule "teacher"
when
  A() implies B()
then
  inject("pupil")
end

rule "pupil"
when
  A()
then
  B()
end
```



Hybrid rules

```
$t : Toss( result == "heads" )
 and @[ kind="Product" args="mixed" ]
$p : Prize( this is "valuable" )
```



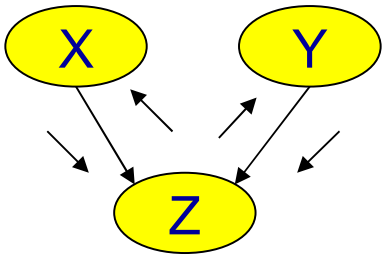
Probability Degree

Fuzzy Degree

Fuzzy Degree
(also the expected value)

Bayesian Networks

Variables → Objects
CPT → Degrees




```
rule "lambda:(Z,X)->Y"
when
 Z() and @[kind="BN"] X()
then
 inject( new Y(),"Y")
end

rule "lambda:(Z,Y)->X"
when
 Z() and @[kind="BN"] Y()
then
 inject( new X(),"X")
end
```

```
rule "pi:(X,Y)->Z"
  degree [0.9 0.8 0.7 0.6 ;
          0.1 0.2 0.3 0.4]
when
  X() or @[kind="BN"] Y()
then
  inject(new Z(),"Z")
end
```

Drools | Questions?

- Dave Bowman: All right, HAL; I'll go in through the emergency airlock.
- HAL: Without your space helmet, Dave, you're going to find that rather difficult.
- Dave Bowman: HAL, I won't argue with you anymore! Open the doors!
- HAL: Dave, this conversation can serve no purpose anymore. Goodbye.

Joshua: **Greetings, Professor Falken.**
Stephen Falken: **Hello, Joshua.**
Joshua: **A strange game. The only winning move is not to play. How about a nice game of chess?**