

# DROOLS

Davide Sottara  
dsotty AT gmail.com

May 28, 2010

# Outline

- 1 Introduzione - Drools
- 2 Richiami di teoria...
- 3 Drools - Linguaggio
- 4 Algoritmo RETE
  - Creazione della RETE
  - A Runtime...
- 5 Riferimenti

# Outline

- 1 Introduzione - Drools
- 2 Richiami di teoria...
- 3 Drools - Linguaggio
- 4 Algoritmo RETE
  - Creazione della RETE
  - A Runtime...
- 5 Riferimenti

# Drools - “Business Logic Integration Platform”



Expert - Rule Engine



Fusion - Event Processing



Flow - Workflow



Guvnor - Rule Repository



Planner - Constraint Solver

# Drools Expert

- Open Source
- Interamente realizzato in Java
- Linguaggio Proprietario (standard previsti...)
- Integrato con Eclipse

# Contatti

Comunità aperta di utenti e sviluppatori

- <http://www.jboss.org/drools>
- <http://blog.athico.com/>
- [irc.codehaus.org](http://irc.codehaus.org) *#drools*

# Outline

- 1 Introduzione - Drools
- 2 Richiami di teoria...
- 3 Drools - Linguaggio
- 4 Algoritmo RETE
  - Creazione della RETE
  - A Runtime...
- 5 Riferimenti

## Dal “Modus Ponens”

$$\frac{\langle P(x), P(X) \rightarrow C(Y) \rangle}{C(y)}$$

- Data l'**implicazione**  $\rightarrow$  (vera)
- la **premessa** (vera) consente di dedurre la **conseguenza** (vera)

## ... alle "Production Rules"

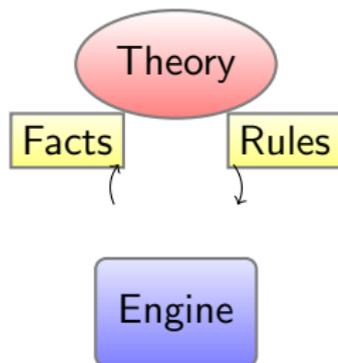
Premise  $\Rightarrow$  Conclusion

- Regola  $\Leftrightarrow$  Implicazione (vera)
- Forward Chaining
- Regole *reattive*

## ... alle "Production Rules"

Premise  $\Rightarrow$  Conclusion

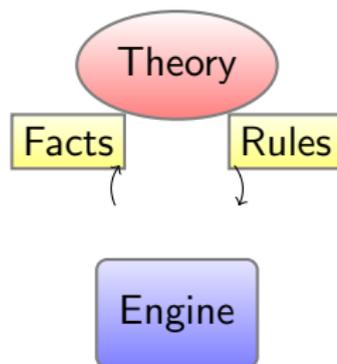
- Regola  $\Leftrightarrow$  Implicazione (vera)
- Forward Chaining
- Regole *reattive*



# ... alle "Production Rules"

Premise  $\Rightarrow$  Conclusion

- Regola  $\Leftrightarrow$  Implicazione (vera)
- Forward Chaining
- Regole *reattive*

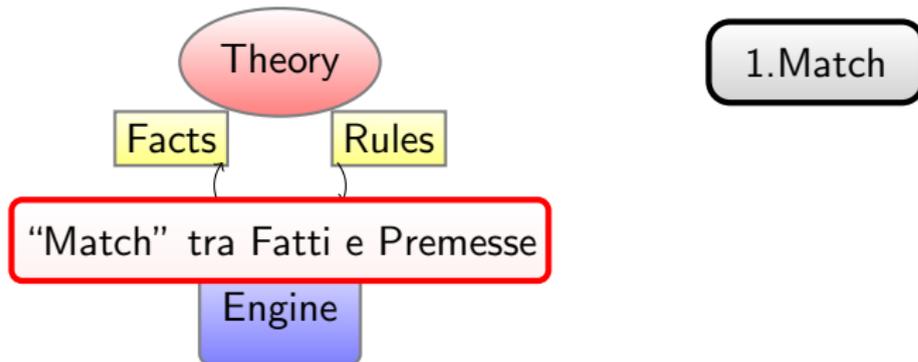


1.Match

## ... alle "Production Rules"

Premise  $\Rightarrow$  Conclusion

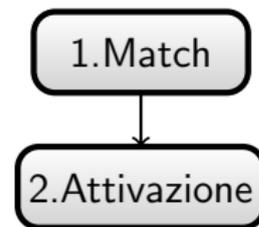
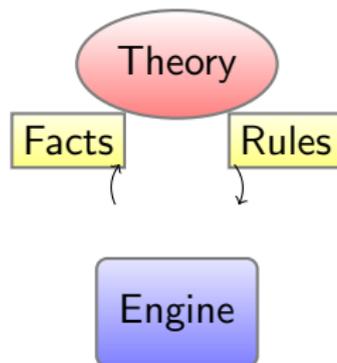
- Regola  $\Leftrightarrow$  Implicazione (vera)
- Forward Chaining
- Regole *reattive*



## ... alle "Production Rules"

Premise  $\Rightarrow$  Conclusion

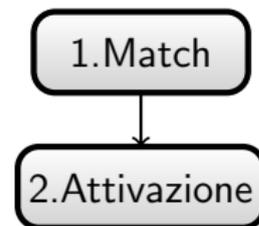
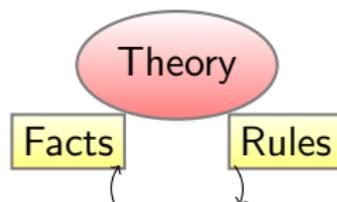
- Regola  $\Leftrightarrow$  Implicazione (vera)
- Forward Chaining
- Regole *reattive*



## ... alle "Production Rules"

Premise  $\Rightarrow$  Conclusion

- Regola  $\Leftrightarrow$  Implicazione (vera)
- Forward Chaining
- Regole *reattive*



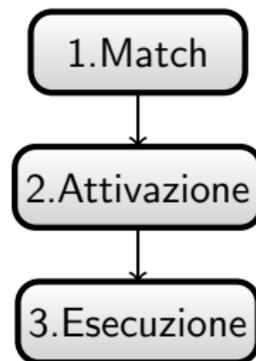
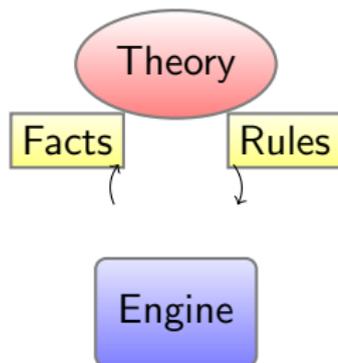
Engine

Regole *attive* accodate in **Agenda**

## ... alle "Production Rules"

Premise  $\Rightarrow$  Conclusion

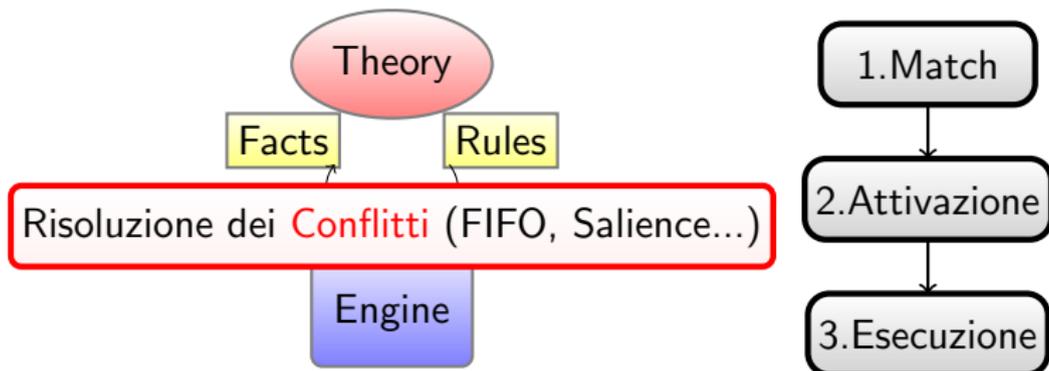
- Regola  $\Leftrightarrow$  Implicazione (vera)
- Forward Chaining
- Regole *reattive*



## ... alle "Production Rules"

Premise  $\Rightarrow$  Conclusion

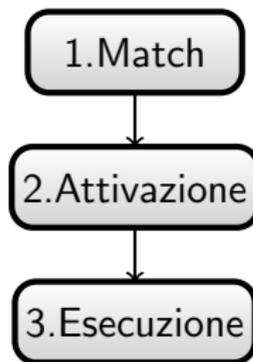
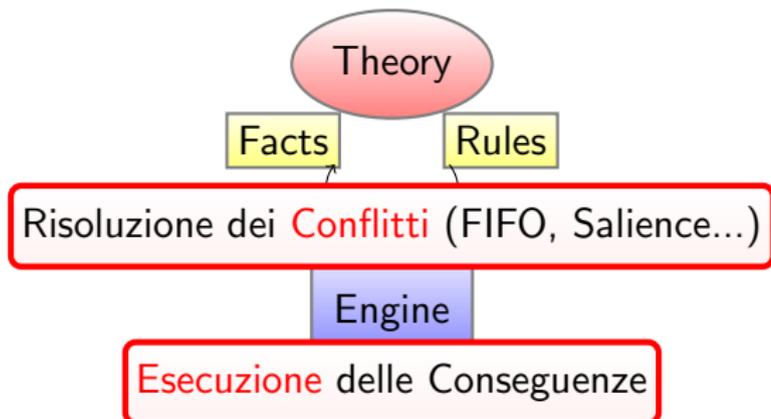
- Regola  $\Leftrightarrow$  Implicazione (vera)
- Forward Chaining
- Regole *reattive*



# ... alle "Production Rules"

Premise  $\Rightarrow$  Conclusion

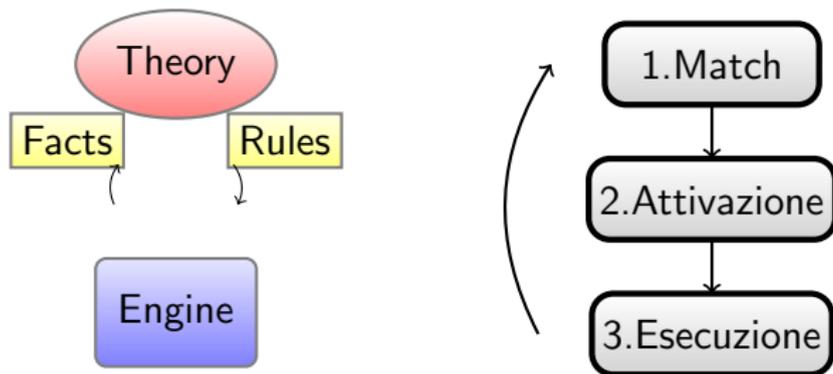
- Regola  $\Leftrightarrow$  Implicazione (vera)
- Forward Chaining
- Regole *reattive*



## ... alle "Production Rules"

Premise  $\Rightarrow$  Conclusion

- Regola  $\Leftrightarrow$  Implicazione (vera)
- Forward Chaining
- Regole *reattive*



# Outline

- 1 Introduzione - Drools
- 2 Richiami di teoria...
- 3 Drools - Linguaggio**
- 4 Algoritmo RETE
  - Creazione della RETE
  - A Runtime...
- 5 Riferimenti

# Regole

Struttura delle regole:

```
rule "ID_Regola"  
    // attributi  
when  
    // LHS – Premessa  
then  
    // RHS – Conclusione  
end
```

# LHS - Pattern / 1

## *Pattern*

- Elemento *atomico* per la scrittura di regole
- Filtra gli oggetti inseriti nella WM
- Definisce un insieme di *vincoli*

# LHS - Pattern / 2

Person( )

- Pattern di base - semplice vincolo di tipo
- `x instanceof Person ?`

## LHS - Pattern / 2

```
Person( name == "john" , age > 18 )
```

- *Field constraints*
- Operatori classici : ==, <, >=, ...
  - in *and* : ',' oppure '&&'
  - in *or* : '||'
- `x.getName().equals("john") && x.getAge() > 18 ?`

## LHS - Pattern / 2

```
$p: Person( $n : name == "john" , age > 18 , $add : address )
```

- *Variabili*
- Assegnamento mediante ':'

# LHS - Join

```
rule "Join"  
when  
    $p : Person( $n : name, age > 30 )  
    Course( $s : subject == "ai", teacher == $p )  
then  
    ...  
end
```

- Pattern multipli nella premessa
- Generano tutte le combinazioni usando oggetti che fanno match
- Variabili per definire constraint *tra* pattern diversi
- *rightarrow* Join nei DB relazionali

# RHS - Conseguenze

## Conseguenze

- Logiche...
  - **Insert** : genera nuovi fatti nella WM
  - **Retract** : rimuove fatti esistenti
  - **Modify** : aggiorna fatti esistenti
- ... e non
  - Qualsiasi "side effect"
  - Codice JAVA libero

# Outline

- 1 Introduzione - Drools
- 2 Richiami di teoria...
- 3 Drools - Linguaggio
- 4 Algoritmo RETE**
  - Creazione della RETE
  - A Runtime...
- 5 Riferimenti

# RETE

- Efficiente per grandi quantità di regole e fatti
- Usato nella maggior parte dei Production Rule System commerciali e non
  - Drools, CLIPS, Jess, ILOG, ...

- Le regole vengono *compile* in una “**data-flow network**”
- I fatti sono memorizzati in una **memoria distribuita**
- I nodi della rete vengono **condivisi** quando possibile

# Outline

- 1 Introduzione - Drools
- 2 Richiami di teoria...
- 3 Drools - Linguaggio
- 4 Algoritmo RETE**
  - Creazione della RETE
  - A Runtime...
- 5 Riferimenti

# RETE al lavoro

## Regola "Professori e Corsi di AI"

```
$p : Professor( $n : name, age > 30 )  
Course( $s : subject == "ai", teacher == $p )
```



# RETE al lavoro

## Regola "Professori e Corsi di AI"

```
$p : Professor( $n : name, age > 30 )  
Course( $s : subject == "ai", teacher == $p )
```

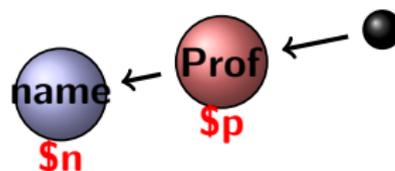


for each Pattern  $j$  : new type node

# RETE al lavoro

## Regola "Professori e Corsi di AI"

$\$p$  : Professor(  $\$n$  : name, age > 30 )  
Course(  $\$s$  : subject == "ai", teacher ==  $\$p$  )

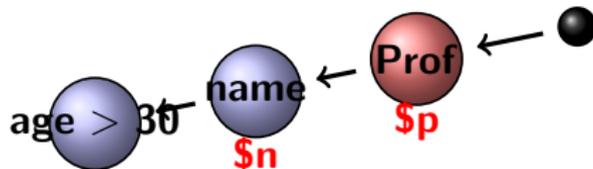


$P_j$  : for each Literal Constraint  $k$  : new alpha node  $\alpha_{j,k}$

# RETE al lavoro

## Regola "Professori e Corsi di AI"

$\$p$  : Professor(  $\$n$  : name, age > 30 )  
Course(  $\$s$  : subject == "ai", teacher ==  $\$p$  )

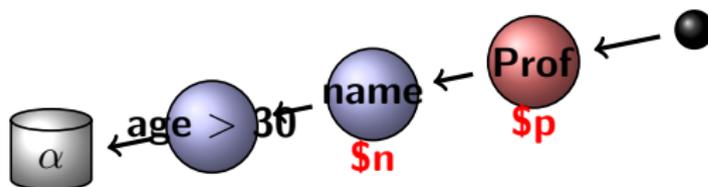


$P_j$  : for each Literal Constraint  $k$  : new alpha node  $\alpha_{j,k}$

# RETE al lavoro

## Regola "Professori e Corsi di AI"

$\$p$  : Professor(  $\$n$  : name, age > 30 )  
Course(  $\$s$  : subject == "ai", teacher ==  $\$p$  )

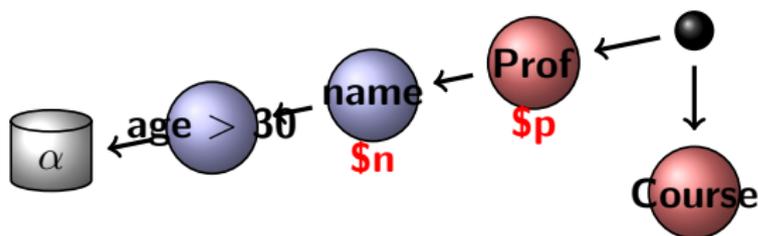


$P_j$  : new alpha memory  $\alpha_j$

# RETE al lavoro

## Regola "Professori e Corsi di AI"

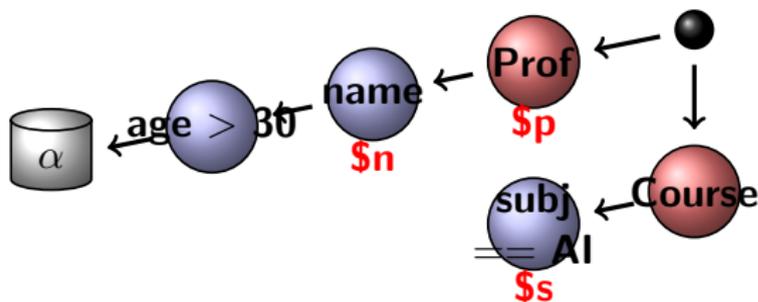
$\$p$  : Professor(  $\$n$  : name, age > 30 )  
Course(  $\$s$  : subject == "ai", teacher ==  $\$p$  )



# RETE al lavoro

## Regola "Professori e Corsi di AI"

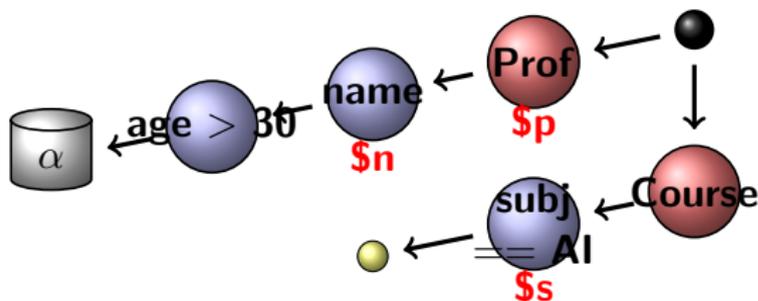
$\$p$  : Professor(  $\$n$  : name, age > 30 )  
Course(  $\$s$  : subject == "ai", teacher ==  $\$p$  )



# RETE al lavoro

## Regola "Professori e Corsi di AI"

$\$p$  : Professor(  $\$n$  : name, age > 30 )  
Course(  $\$s$  : subject == "ai", teacher ==  $\$p$  )

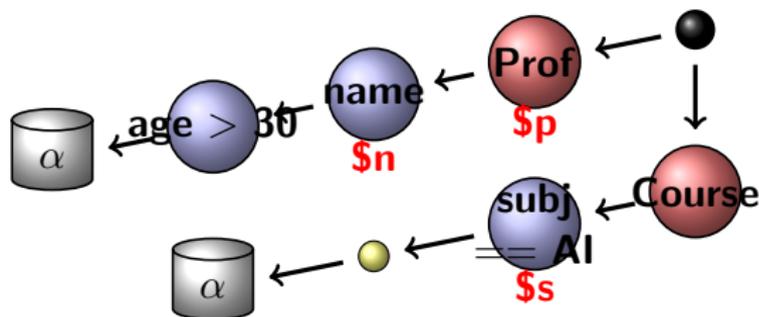


(Skip Variable Constraints)

# RETE al lavoro

## Regola "Professori e Corsi di AI"

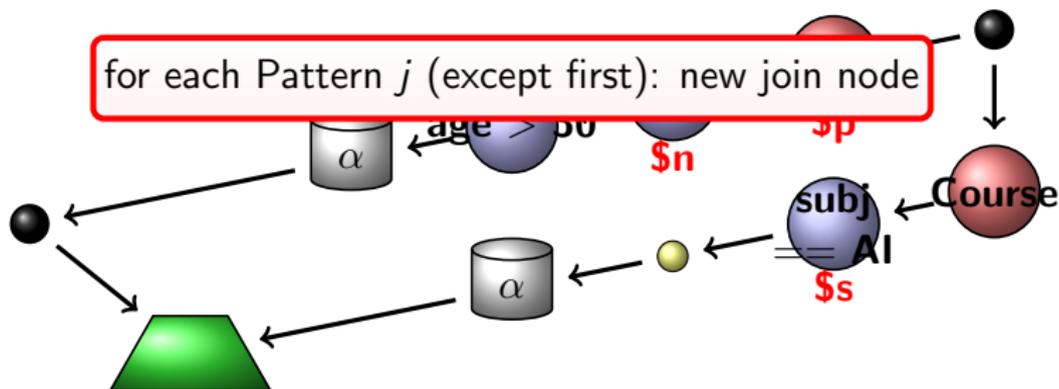
$\$p$  : Professor(  $\$n$  : name, age > 30 )  
Course(  $\$s$  : subject == "ai", teacher ==  $\$p$  )



# RETE al lavoro

## Regola "Professori e Corsi di AI"

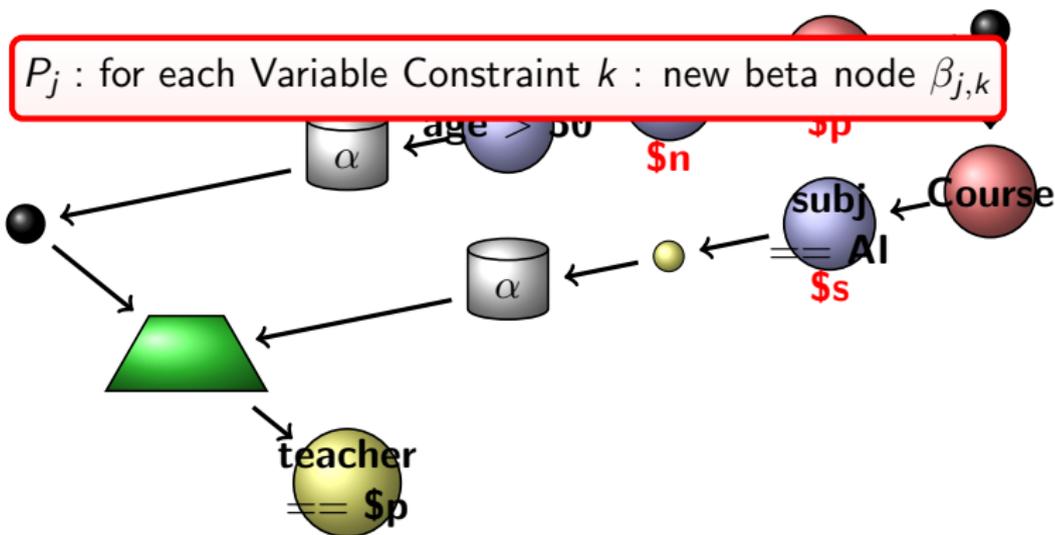
$\$p$  : Professor(  $\$n$  : name, age > 30 )  
 Course(  $\$s$  : subject == "ai", teacher ==  $\$p$  )



# RETE al lavoro

## Regola "Professori e Corsi di AI"

$\$p$  : Professor(  $\$n$  : name, age > 30 )  
 Course(  $\$s$  : subject == "ai", teacher ==  $\$p$  )

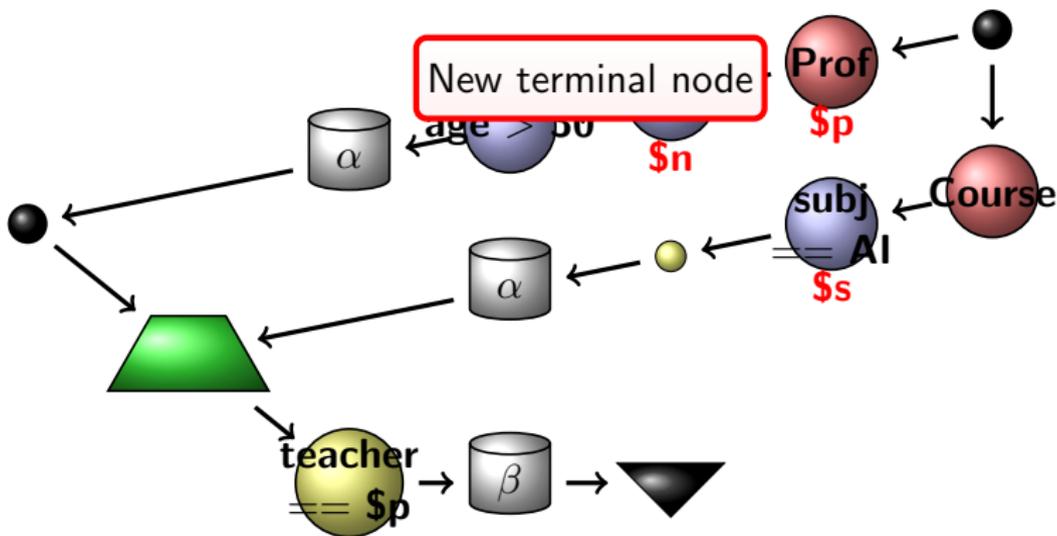




# RETE al lavoro

## Regola "Professori e Corsi di AI"

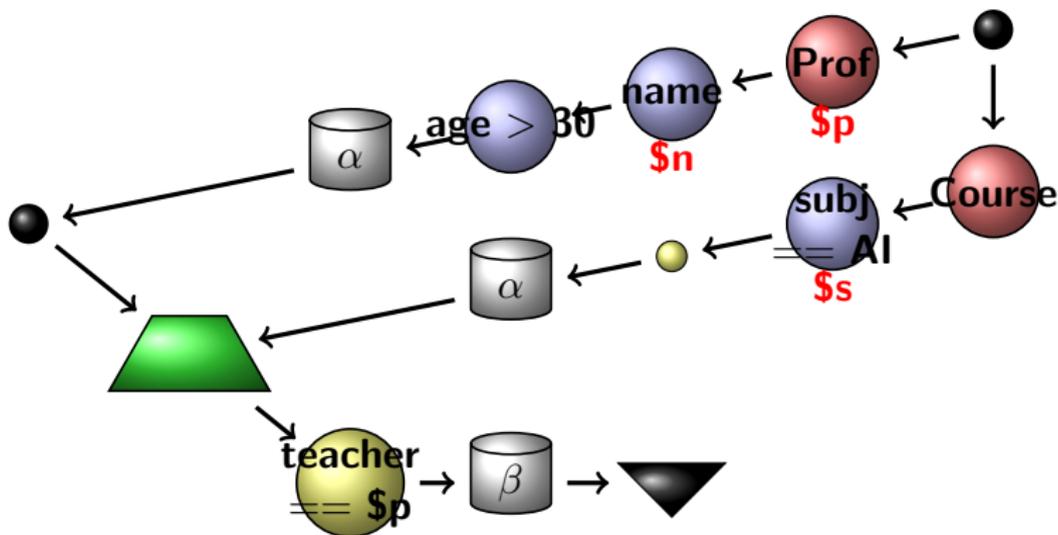
\$p : Professor( \$n : name, age > 30 )  
 Course( \$s : subject == "ai", teacher == \$p )



# RETE al lavoro

## Regola "Professori e Corsi di AI"

\$p : Professor( \$n : name, age > 30 )  
 Course( \$s : subject == "ai", teacher == \$p )



# Outline

- 1 Introduzione - Drools
- 2 Richiami di teoria...
- 3 Drools - Linguaggio
- 4 Algoritmo RETE**
  - Creazione della RETE
  - A Runtime...**
- 5 Riferimenti

# Nodi:

## Alpha Network

- I nodi “Type” selezionano i fatti (vincolo sulla classe)
- I nodi “Alpha” filtrano i fatti (vincolo sui campi)
- I fatti non scartati vengono salvati nelle “alpha-memories”
- I fatti nelle “alpha-memories” fanno *match* con un pattern

## Beta Network

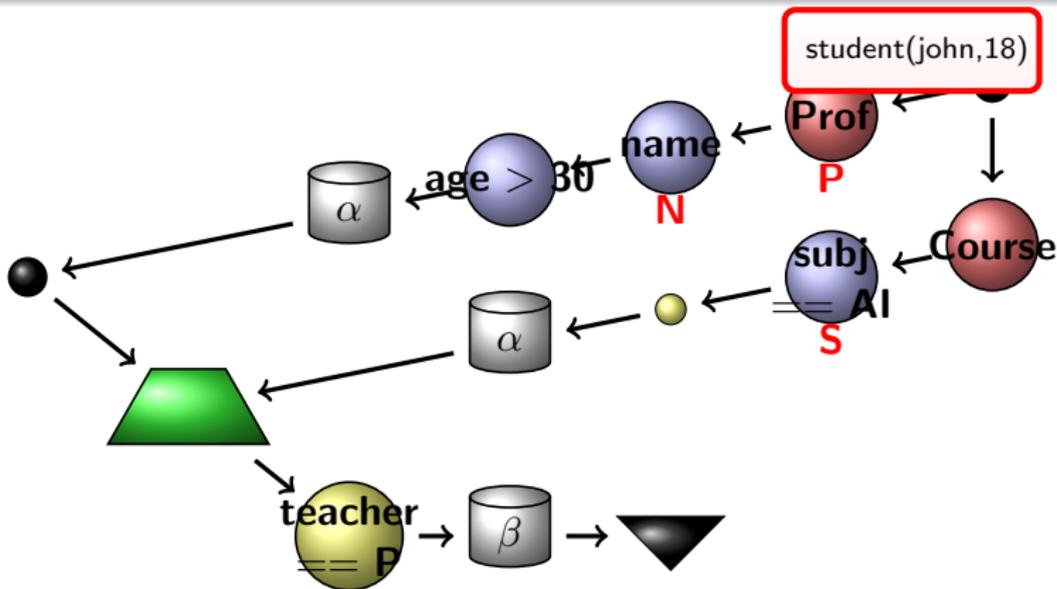
- I nodi “Join” collegano memorie alpha e beta
- I nodi “Join” creano **n-tuple** a partire da fatti e (n-1)-tuple
- I nodi “Beta” filtrano le tuple
- Le tuple non scartate vengono salvate nelle “beta-memories”
- Le tuple nelle “beta-memories” fanno *match* con una sequenza (parziale) di pattern

A Runtime...

## A runtime...

## Regola "Professori e Corsi di AI"

$\$p$  : Professor(  $\$n$  : name, age > 30 )  
 Course(  $\$s$  : subject == "ai", teacher ==  $\$p$  )

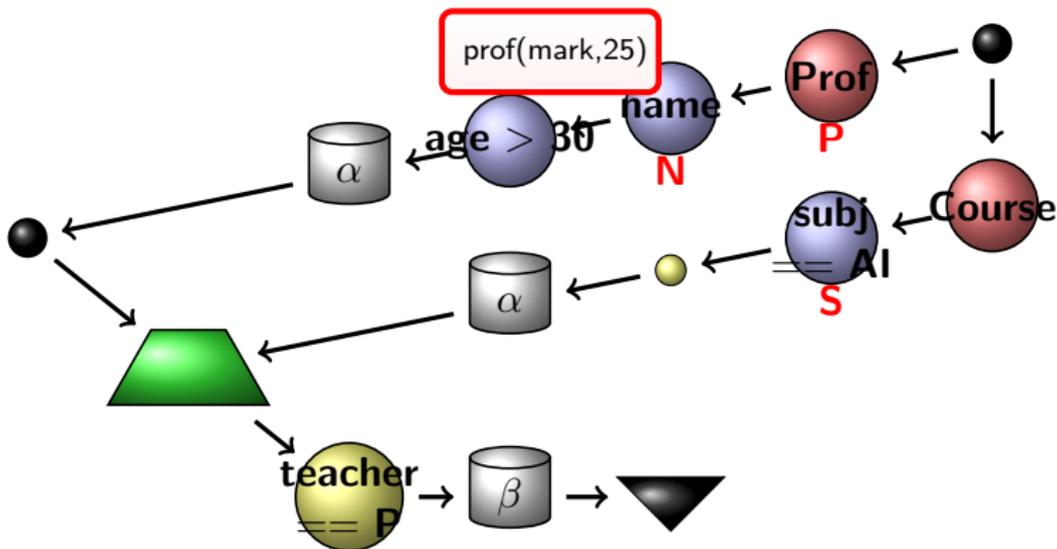


A Runtime...

## A runtime...

## Regola "Professori e Corsi di AI"

$\$p$  : Professor(  $\$n$  : name, age > 30 )  
 Course(  $\$s$  : subject == "ai", teacher ==  $\$p$  )

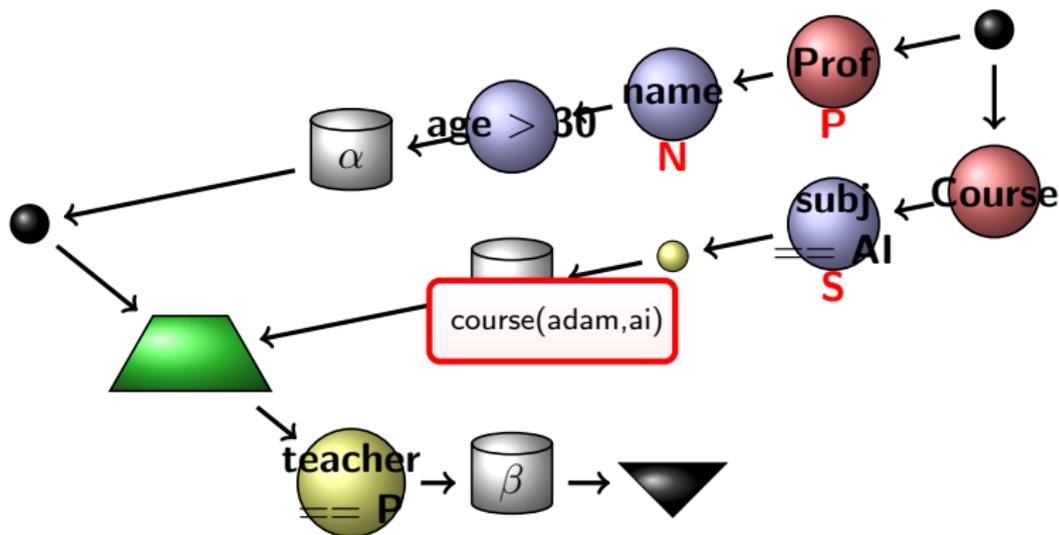


A Runtime...

## A runtime...

## Regola "Professori e Corsi di AI"

\$p : Professor( \$n : name, age > 30 )  
 Course( \$s : subject == "ai", teacher == \$p )

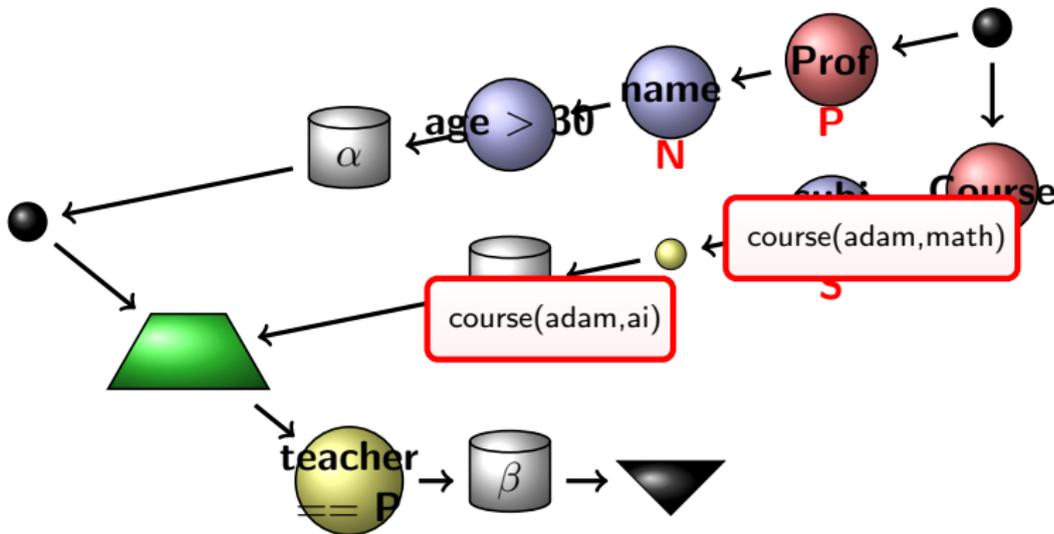


A Runtime...

## A runtime...

## Regola "Professori e Corsi di AI"

\$p : Professor( \$n : name, age > 30 )  
 Course( \$s : subject == "ai", teacher == \$p )

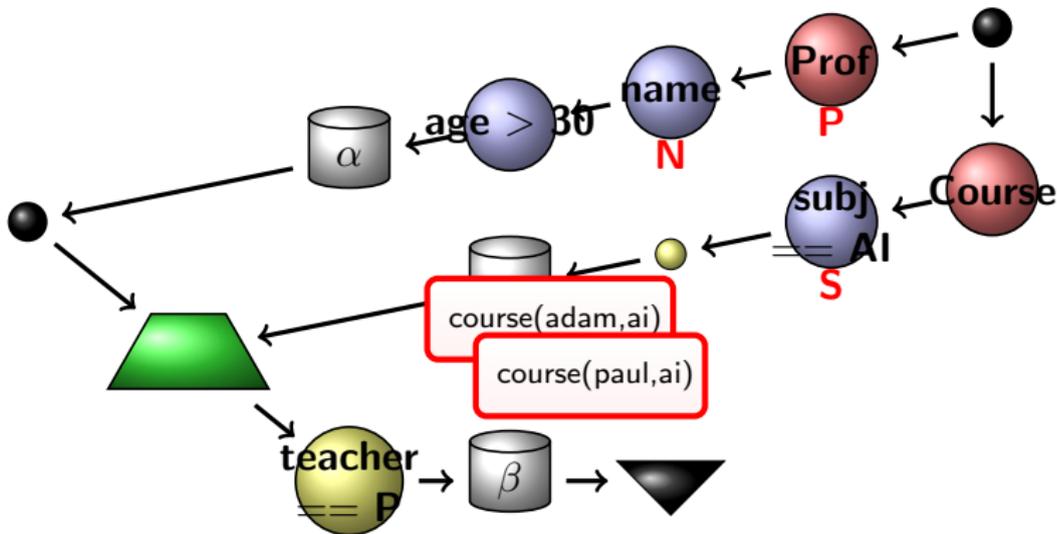


A Runtime...

## A runtime...

## Regola "Professori e Corsi di AI"

\$p : Professor( \$n : name, age > 30 )  
 Course( \$s : subject == "ai", teacher == \$p )

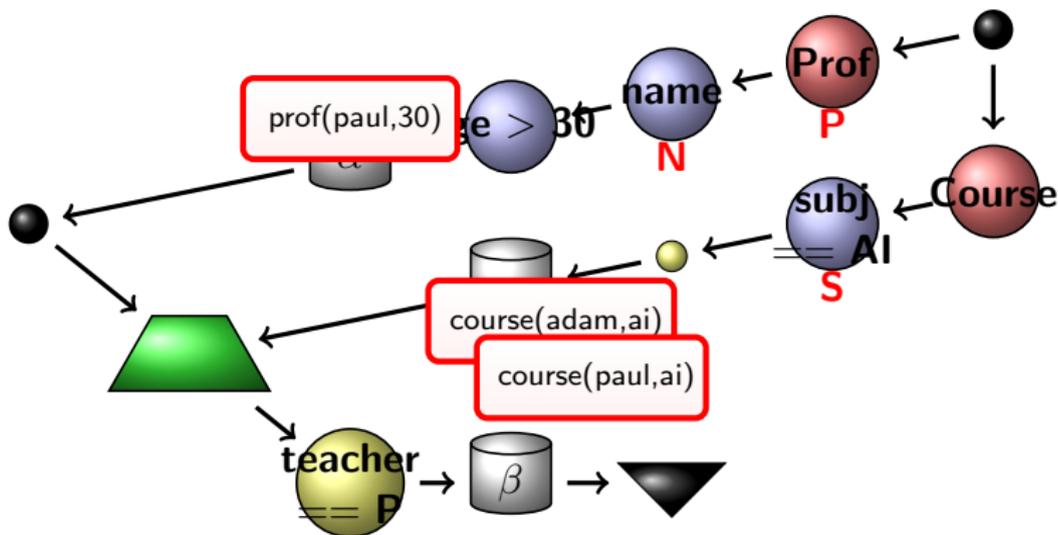


A Runtime...

## A runtime...

## Regola "Professori e Corsi di AI"

\$p : Professor( \$n : name, age > 30 )  
 Course( \$s : subject == "ai", teacher == \$p )

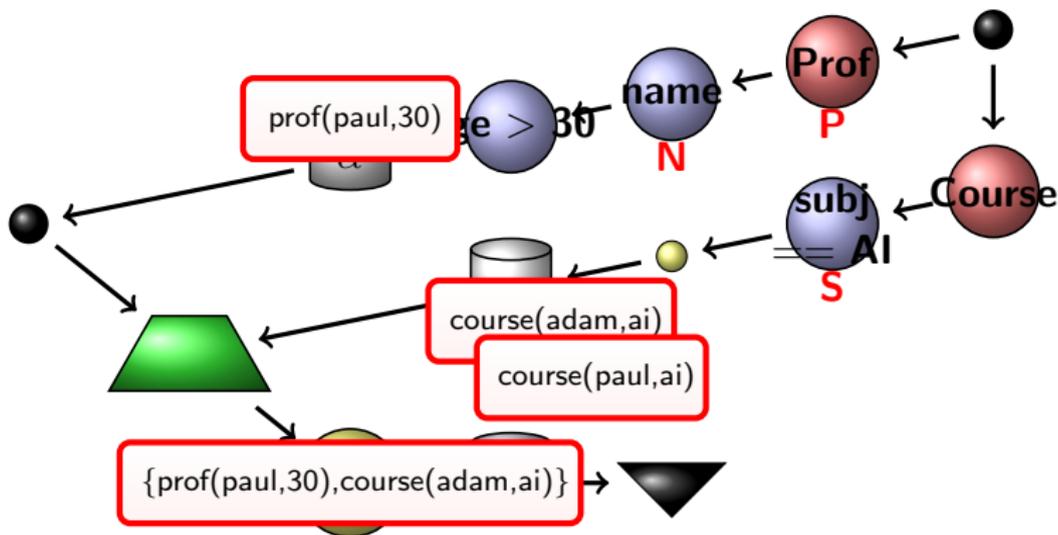


A Runtime...

## A runtime...

## Regola "Professori e Corsi di AI"

\$p : Professor( \$n : name, age > 30 )  
 Course( \$s : subject == "ai", teacher == \$p )

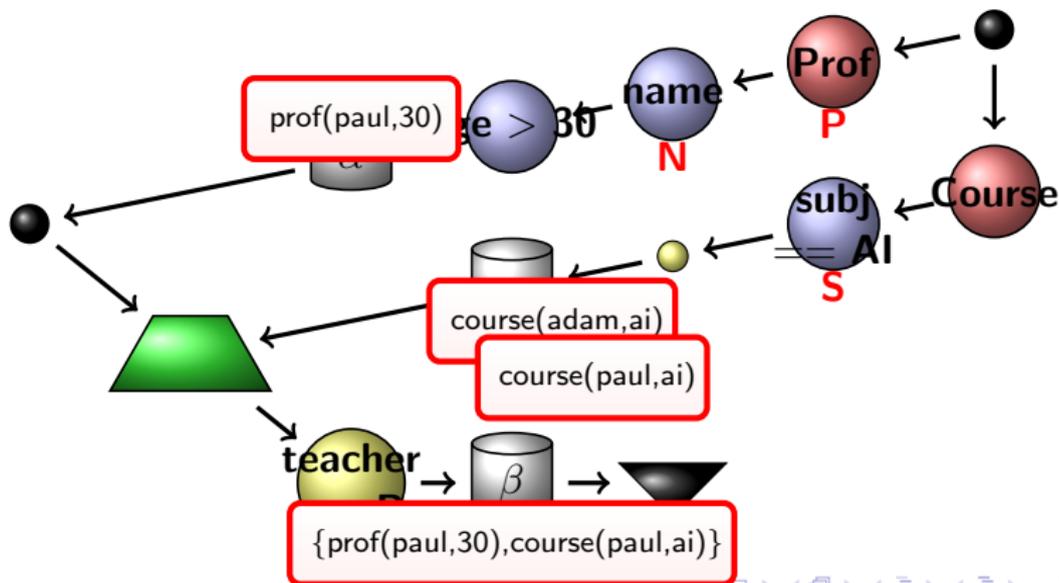


A Runtime...

## A runtime...

## Regola "Professori e Corsi di AI"

\$p : Professor( \$n : name, age > 30 )  
 Course( \$s : subject == "ai", teacher == \$p )





# Outline

- 1 Introduzione - Drools
- 2 Richiami di teoria...
- 3 Drools - Linguaggio
- 4 Algoritmo RETE
  - Creazione della RETE
  - A Runtime...
- 5 Riferimenti

## Link utili

- Charles Forgy, “Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem”, Artificial Intelligence, 19, pp 17-37, 1982
- R.B. Doorenbos, “Production Matching for Large Learning Systems”, [www.zionis.org/CMU-CS-95-113.pdf](http://www.zionis.org/CMU-CS-95-113.pdf)
- [http://en.wikipedia.org/wiki/Rete\\_algorithm](http://en.wikipedia.org/wiki/Rete_algorithm)
  
- Sito : <http://www.jboss.org/drools>
- Blog : <http://blog.athico.com/>
- IRC : [#drools](http://irc.codehaus.org)