Università di Bergamo
Facoltà di Ingegneria

Intelligenza Artificiale

Paolo Salvaneschi

B1_3 V1.2

Introduzione a JESS

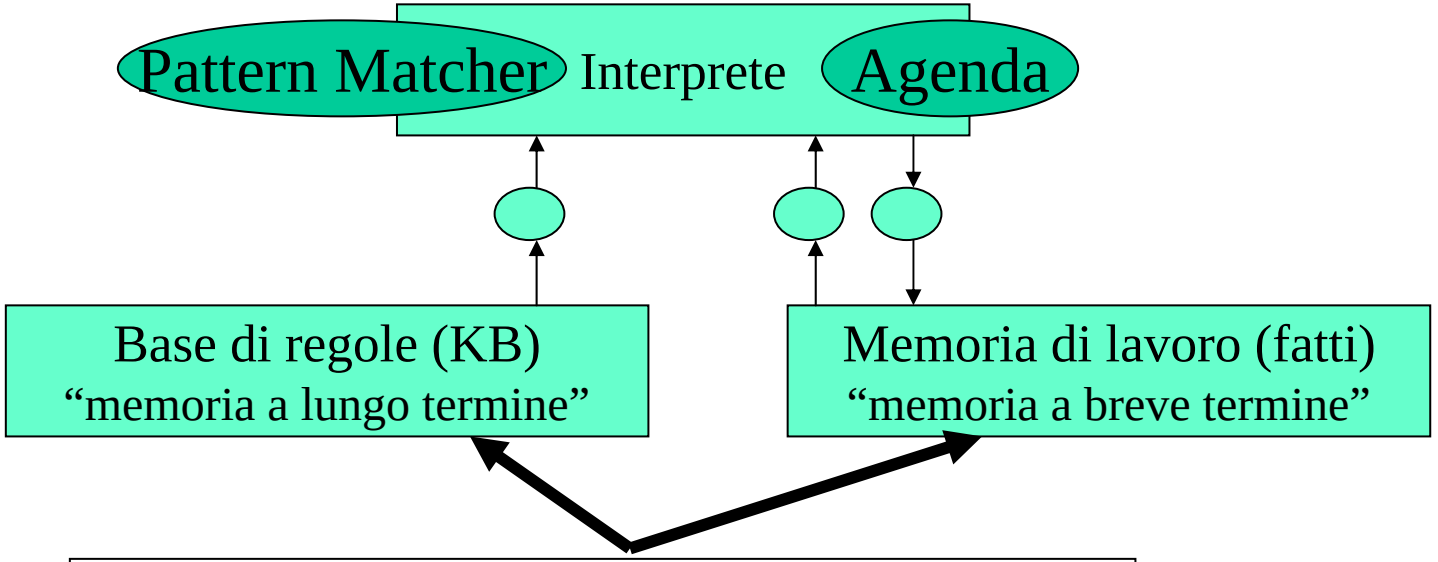
Parte B - Note

Il contenuto del documento è liberamente utilizzabile dagli studenti, per studio personale e per supporto a lezioni universitarie.
Ogni altro uso è riservato, e deve essere preventivamente autorizzato dall' autore.
Sono graditi commenti o suggerimenti per il miglioramento del materiale

INDICE

- Jess, Sistemi a regole e FOL
- KB in clausole di Horn
- Il problema della negazione
- Mantenimento della verità

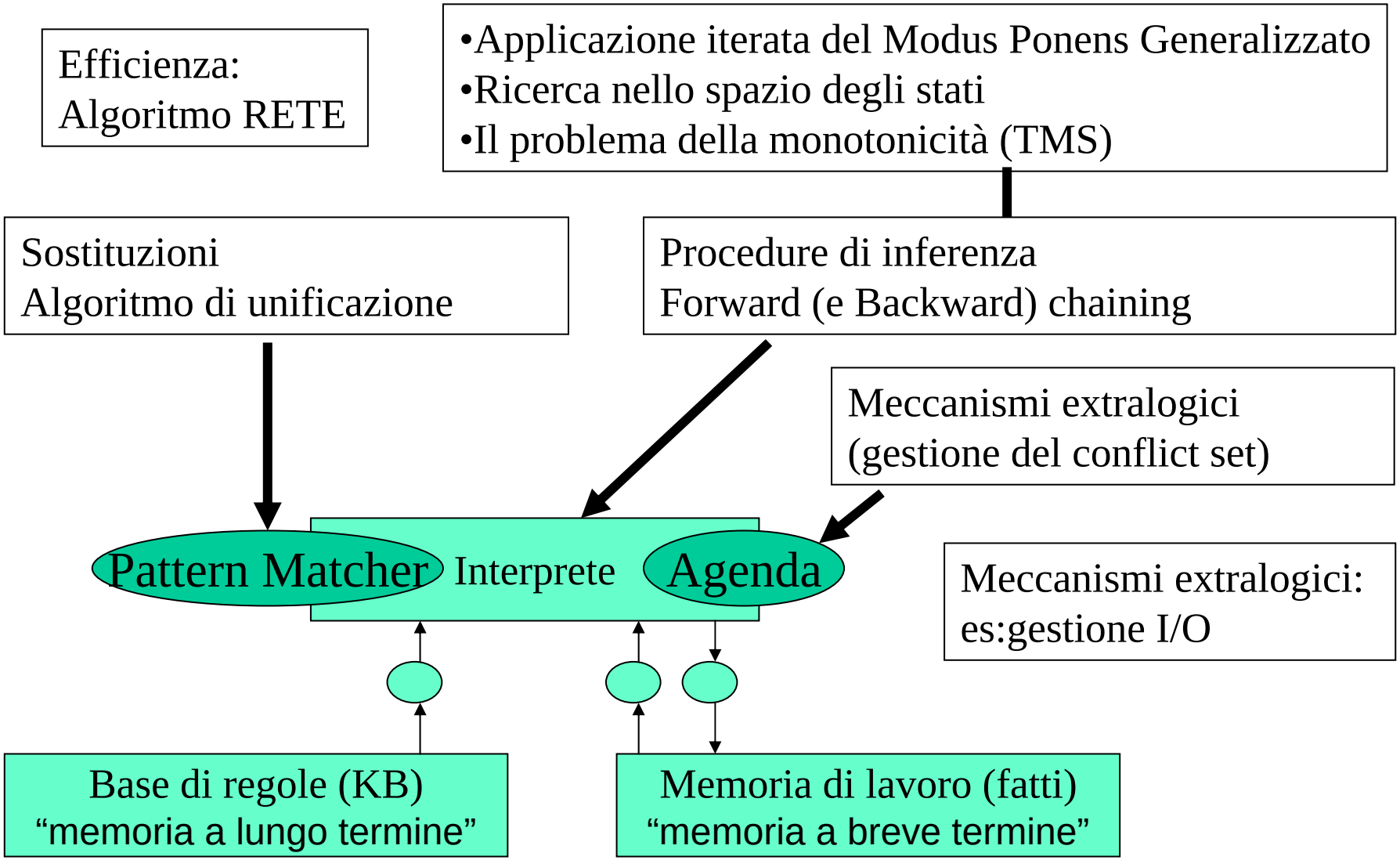
Jess, Sistemi a regole e FOL



KB in definite horn clauses (restrizione di FOL)

- KB facile da leggere
- Algoritmi di inferenza forward e backward chaining semplici
- Complessità per decidere l’implicazione logica lineare rispetto alle dimensioni di KB

Jess, Sistemi a regole e FOL



- Equivalenze logiche

$$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha) \quad \text{commutativity of } \wedge$$
$$(\alpha \vee \beta) \equiv (\beta \vee \alpha) \quad \text{commutativity of } \vee$$
$$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma)) \quad \text{associativity of } \wedge$$
$$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma)) \quad \text{associativity of } \vee$$
$$\neg(\neg\alpha) \equiv \alpha \quad \text{double-negation elimination}$$
$$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha) \quad \text{contraposition}$$
$$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta) \quad \text{implication elimination}$$
$$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination}$$
$$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta) \quad \text{de Morgan}$$
$$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta) \quad \text{de Morgan}$$
$$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) \quad \text{distributivity of } \wedge \text{ over } \vee$$
$$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \quad \text{distributivity of } \vee \text{ over } \wedge$$

KB in clausole di Horn

- Clausole di Horn
- Le basi di conoscenza reali possono contenere clausole appartenenti ad una classe con una restrizione chiamate Clausole di Horn. Per esse esiste una procedura di inferenza in tempo polinomiale.
- Clausole di Horn:

  Disgiunzione di letterali di cui <u>al massimo</u> uno è positivo

$(\neg A \vee \neg B \vee C)$ è una clausola di Horn

$(\neg B \vee C \vee D)$ Non è una clausola di Horn

KB in clausole di Horn

- Utilizzo delle Clausole di Horn
- Ogni clausola di Horn può essere scritta come una implicazione la cui premessa è una congiunzione di letterali positivi e la cui conclusione è un unico letterale positivo
- Modo facile da leggere usato nelle basi di conoscenza

$(A \wedge B \Rightarrow C)$ Regola

(Se lo spostamento K1 è intermedio
e lo spostamento K2 è elevato
allora lo stato è preallarme)

KB in clausole di Horn

- Utilizzo delle Clausole di Horn
- L'inferenza con le clausole di Horn può essere eseguita applicando il Modus Ponens ovunque possibile, fino a quando non si può eseguire alcuna inferenza.
- L'inferenza con le clausole di Horn può essere eseguita con gli algoritmi forward chaining e backward chaining. Facili da capire
- Complessità per decidere l'implicazione logica: lineare rispetto alle dimensioni di KB

KB in clausole di Horn

- Clausole di Horn: Disgiunzione di letterali di cui al massimo uno è positivo
- Definite clauses (base per la programmazione logica): Disgiunzione di letterali di cui esattamente uno è positivo

$$(\underbrace{\neg A \vee \neg B}_{\text{body}} \vee \underbrace{C}_{\text{head}})$$

E' una definite Horn clause

$$(C)$$

fact

$$(\neg A \vee \neg B)$$

E' una Horn clause
Non è una definite Horn clause

KB in clausole di Horn

- Algoritmi di Forward-chaining e Backward-chaining
- Inferenza su una base di conoscenza costituita da una congiunzione di Clausole di Horn
- Complessità lineare rispetto alle dimensioni di KB
- Es di KB: $C \wedge (B \Rightarrow A) \wedge (C \wedge D \Rightarrow B)$

KB in clausole di Horn

- Come si traduce in clausole definite di Horn

Caso 1--- RHS costituito da più assert

$$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha) \quad \text{commutativity of } \wedge$$
$$(\alpha \vee \beta) \equiv (\beta \vee \alpha) \quad \text{commutativity of } \vee$$
$$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma)) \quad \text{associativity of } \wedge$$
$$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma)) \quad \text{associativity of } \vee$$
$$\neg(\neg\alpha) \equiv \alpha \quad \text{double-negation elimination}$$
$$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha) \quad \text{contraposition}$$
$$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta) \quad \text{implication elimination}$$
$$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination}$$
$$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta) \quad \text{de Morgan}$$
$$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta) \quad \text{de Morgan}$$
$$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) \quad \text{distributivity of } \wedge \text{ over } \vee$$
$$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \quad \text{distributivity of } \vee \text{ over } \wedge$$

$$A \Rightarrow B \wedge C$$

$$\neg A \vee (B \wedge C)$$ eliminazione dell'implicazione

$$(\neg A \vee B) \wedge (\neg A \vee C)$$ distributività di $\vee$ su $\wedge$

$$A \Rightarrow B \wedge A \Rightarrow C$$ eliminazione dell'implicazione

La regola è tradotta in due regole con LHS originale identico e un congiunto di RSH originale ognuna

KB in clausole di Horn

- Come si traduce in clausole definite di horn

Caso 2--- LHS utilizza OR

$$A \vee B \Rightarrow C$$

$$\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{de Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{de Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}$$

$\neg (A \vee B) \vee C$ eliminazione dell'implicazione

$(\neg A \wedge \neg B) \vee C$ De Morgan

$(\neg A \vee C) \wedge (\neg B \vee C)$ distributività di $\vee$ su $\wedge$

$A \Rightarrow C \wedge B \Rightarrow C$ eliminazione dell'implicazione

La regola è tradotta in due regole ognuna con un disgiunto di LSH originale e identico RHS originale

KB in clausole di Horn

The 'or' conditional element.

Any number of patterns can be enclosed in a list with OR as the head.
The resulting pattern is matched if one or more of the patterns inside the or are matched. If more than one of the subpatterns are matched, the or is matched more than once

```
Jess> (defrule or-example-1
      (or (a) (b) (c))
      =>)
Jess> (assert (a) (b) (c))
Jess> (printout t (run) crlf)
3
```

Restituisce il numero di regole accese

La regola è attivata tre volte

KB in clausole di Horn

Subrule generation and the 'or' conditional element.
A rule containing an 'or' conditional element with *n* branches is precisely equivalent to *n* rules, each of which has one branch as its left hand side. In fact, this is how the or conditional element is implemented: Jess imternally generates one new rule for each branch. Each of these generated rules is a *subrule*.

Note that if the right hand side of a rule uses a variable defined by matching on the left hand side of that rule, and the variable is defined by one or more branches of an or pattern but not all branches, then a runtime error may occur.

KB in clausole di Horn

- Come si traduce in clausole definite di horn

Caso 3 – non si può scrivere in Jess

$$A \Rightarrow B \vee C$$

$$\neg A \vee B \vee C$$ eliminazione dell'implicazione

Non è una clausola di Horn
(Disgiunzione di letterali di cui al massimo uno è positivo)

Il problema della negazione

- Il problema della negazione
- Ogni clausola di Horn può essere scritta come una implicazione la cui premessa è una congiunzione di letterali positivi e la cui conclusione è un unico letterale positivo

  $A \wedge B \Rightarrow C$

  NonPiove $\Rightarrow$ PuoiUscire

  Modo scomodo: sarebbe più comodo usare la negazione in modo esplicito

  not (piove) $\Rightarrow$ PuoiUscire

Il problema della negazione

2.8.6. The 'not' conditional element.

Any single pattern can be enclosed in a list with not as the head. In this case, the pattern is considered to match if a fact (or set of facts) which matches the pattern is *not* found.

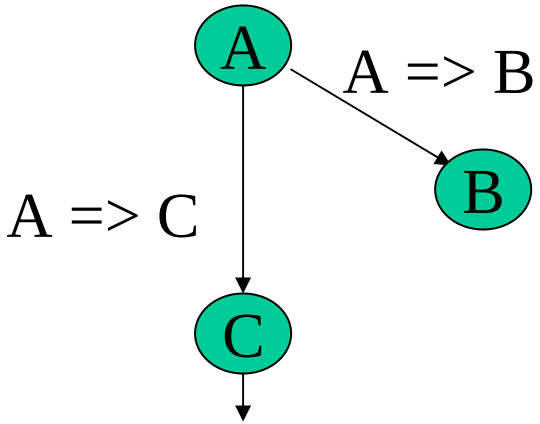
Mantenimento della verità

- Il problema della monotonicità, retract e TMS
  - Cosa accade se la base di conoscenza è non monotona?
  - Ad esempio è retracted un fatto dalla cui verità dipendevano altri fatti (ora non più veri)
  - Non è possibile utilizzare algoritmi forward e backward corretti e completi

Mantenimento della verità

Es_TMS1

```
;;
;; Esercizio TMS1
;;
(reset)
(watch all)
(deffunction change-baby () (printout t "Baby is now dry" crlf))
(defrule sethelp-baby-if-wet
      (baby-is-wet)
      =>
       (assert (sethelp)))
(defrule change-baby-if-wet
      ?wet <- (baby-is-wet)
      =>
      (change-baby)
       (retract ?wet))
(assert (baby-is-wet))
(run)
```



A non è più vero
B resta vero anche
se non lo dovrebbe essere più

Mantenimento della verità

```
Jess> ( batch appoggio/es_TMS1.clp )
 ==> Focus MAIN
 ==> f-0 (MAIN::initial-fact)
MAIN::sethelp-baby-if-wet: =1=1=1+t
MAIN::change-baby-if-wet: =1=1=1+t
 ==> f-1 (MAIN::baby-is-wet)
==> Activation: MAIN::sethelp-baby-if-wet :  f-1
==> Activation: MAIN::change-baby-if-wet :  f-1
FIRE 1 MAIN::sethelp-baby-if-wet f-1
 ==> f-2 (MAIN::sethelp)
FIRE 2 MAIN::change-baby-if-wet f-1
Baby is now dry
 <== f-1 (MAIN::baby-is-wet)
 <== Focus MAIN
2
Jess> (facts)
f-0   (MAIN::initial-fact)
f-2   (MAIN::sethelp)
For a total of 2 facts.
Jess>
```

Mantenimento della verità

- TMS – Truth Maintenance System
  - Sistema che garantisce la coerenza della base di conoscenza ritrattando tutte le implicazioni che dipendevano da un fatto che viene ritrattato.
  - Jess non fornisce un TMS ma un meccanismo di base che deve essere gestito manualmente

Mantenimento della verità

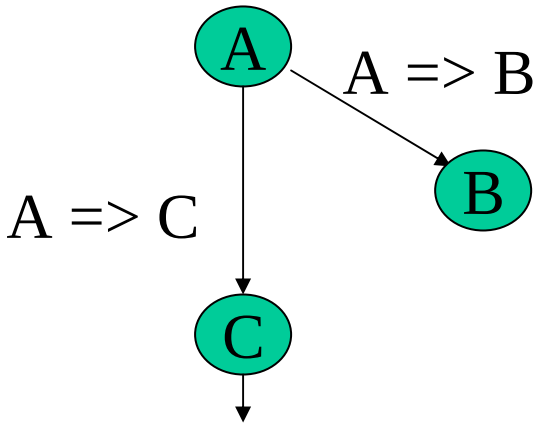
The 'logical' conditional element.

The logical conditional element lets you specify *logical dependencies* among facts. All the facts asserted on the RHS of a rule become dependent on the matches to the logicalpatterns on that rule's LHS. If any of the matches later become invalid, the dependent facts are retracted automatically.

Mantenimento della verità

Es_TMS2

```
;;
;; Esercizio TMS2-- logical
;;
(reset)
(watch all)
(deffunction change-baby () (printout t "Baby is now dry" crlf))
(defrule sethelp-baby-if-wet
      (logical(baby-is-wet))
      =>
       (assert (sethelp)))
(defrule change-baby-if-wet
      ?wet <- (baby-is-wet)
      =>
      (change-baby)
       (retract ?wet))
(assert (baby-is-wet))
(run)
```



A non è più vero
B è ritrattato

Mantenimento della verità

```
Jess> (batch appoggio/es_TMS2.clp)
 ==> Focus MAIN
 ==> f-0 (MAIN::initial-fact)
MAIN::sethelp-baby-if-wet: =1=1=1+t
MAIN::change-baby-if-wet: =1=1=1+t
 ==> f-1 (MAIN::baby-is-wet)
==> Activation: MAIN::sethelp-baby-if-wet :  f-1
==> Activation: MAIN::change-baby-if-wet :  f-1
FIRE 1 MAIN::sethelp-baby-if-wet f-1
 ==> f-2 (MAIN::sethelp)
FIRE 2 MAIN::change-baby-if-wet f-1
Baby is now dry
 <== f-1 (MAIN::baby-is-wet)
 <== f-2 (MAIN::sethelp)
 <== Focus MAIN
2
Jess> (facts)
f-0   (MAIN::initial-fact)
For a total of 1 facts.
Jess>
```