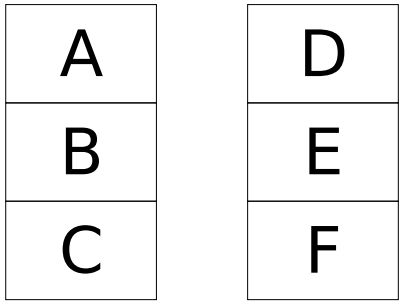
Il mondo dei blocchi

Obiettivo

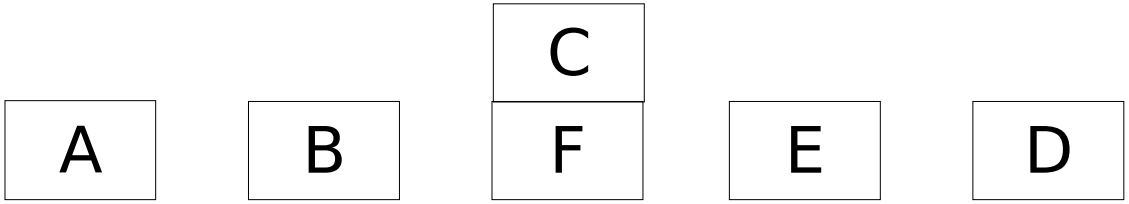
Vogliamo implementare un sistema a regole per la risoluzione del problema del mondo dei blocchi.

Scenario

Supponiamo di avere la seguente configurazione iniziale



e di voler mettere il blocco C su blocco F.



Rappresentazione del dominio

- Ipotesi: i blocchi verranno definiti con dei fatti ordinati (non utilizzeremo un template) del tipo
  (blocco a)
  (blocco b)
  ....
- Rappresentazione della posizione dei blocchi
  - Definire un template per rappresentare la posizione di un blocco
- Rappresentare lo stato finale
  - Utilizzare un template per descrivere lo stato obiettivo

Rappresentazione del dominio (2)

```
(deftemplate sopra-a "Indica che un blocco è su un altro"
   (slot sopra)
   (slot sotto))

(deftemplate obiettivo "Descrive un obiettivo"
   (slot sposta)
   (slot sopra-a))
```

Regola 1

Spostare un blocco libero su un altro blocco libero

un blocco è libero se non ha blocchi sopra

Se

l'obiettivo è di spostare il blocco 1 sul blocco 2 **e**
il blocco 1 è libero **e**
il blocco 2 è libero **e**
il blocco 1 sta sopra il blocco 3

allora

sposta il blocco 1 sul blocco 2

Regola 1

```
(defrule sposta "Sposta un blocco libero su un altro blocco libero"
   ?obiettivo <- (obiettivo (sposta ?blocco1) (sopra-a ?blocco2))
   (blocco ?blocco1)
   (blocco ?blocco2)
   (sopra-a (sopra niente) (sotto ?blocco1))
   ?pila1 <- (sopra-a (sopra ?blocco1) (sotto ?blocco3))
   ?pila2 <- (sopra-a (sopra niente) (sotto ?blocco2))
   =>
   (retract ?obiettivo ?pila1 ?pila2)
   (assert (sopra-a (sopra ?blocco1) (sotto ?blocco2))
           (sopra-a (sopra niente) (sotto ?blocco3)))
   (printout t "Sposta " ?blocco1 " sopra a" ?blocco2 "." crlf))
```

Regola 2

Spostare un blocco libero sul piano

Se

l'obiettivo è di spostare il blocco 1 sul piano **e**
il blocco 1 è libero **e**
il blocco 1 sta sopra il blocco 2 **e**

allora

sposta il blocco 1 sul piano

Regola 2

```
(defrule sposta-sul-piano "Sposta un blocco libero sul piano"
   ?obiettivo <- (obiettivo (sposta ?blocco1) (sopra-a piano))
   (blocco ?blocco1)
   (sopra-a (sopra niente) (sotto ?blocco1))
   ?pila <- (sopra-a (sopra ?blocco1) (sotto ?blocco2))
   =>
   (retract ?obiettivo ?pila)
   (assert (sopra-a (sopra ?blocco1) (sotto piano))
           (sopra-a (sopra niente) (sotto ?blocco2)))
   (printout t "Sposta " ?blocco1 " sul piano." crlf))
```

Regola 3

Stabilire l'obiettivo di liberare il blocco di partenza

Se

l'obiettivo è di spostare il blocco 1 su un altro blocco **e**
il blocco 2 sta sopra il blocco 1

allora

stabilisci l'obiettivo di spostare il blocco 2 sul piano

Regola 3

```
(defrule libera-blocco-partenza
        "Stabilisce l'obiettivo di liberare il blocco di partenza"

   (obiettivo (sposta ?blocco1) (sopra-a ?))  ; Si noti il wild card.

   (blocco ?blocco1)

   (sopra-a (sopra ?blocco2) (sotto ?blocco1))

   (blocco ?blocco2)

   =>

   (assert (obiettivo (sposta ?blocco2) (sopra-a piano))))
```

Regola 4

Stabilire l'obiettivo di liberare il blocco di arrivo

Se

l'obiettivo è di spostare il blocco 1 su un altro blocco **e**
il blocco 2 sta sopra il blocco 1

allora

stabilisci l'obiettivo di spostare il blocco 2 sul piano

Regola 4

```
(defrule libera-blocco-arrivo
        "Stabilisce l'obiettivo di liberare il blocco d'arrivo"

  (obiettivo (sposta ?) (sopra-a ?blocco1))  ; Di nuovo il wild card.

  (blocco ?blocco1)

  (sopra-a (sopra ?blocco2) (sotto ?blocco1))

  (blocco ?blocco2)

  =>

  (assert (obiettivo (sposta ?blocco2) (sopra-a piano))))
```

Fatti

```
(deffacts stato-iniziale "A/B/C, D/E/F, e vogliamo mettere C su F."
   (blocco A)
   (blocco B)
   (blocco C)
   (blocco D)
   (blocco E)
   (blocco F)
   (sopra-a (sopra niente) (sotto A))
   (sopra-a (sopra A) (sotto B))
   (sopra-a (sopra B) (sotto C))
   (sopra-a (sopra C) (sotto piano))
   (sopra-a (sopra niente) (sotto D))
   (sopra-a (sopra D) (sotto E))
   (sopra-a (sopra E) (sotto F))
   (sopra-a (sopra F) (sotto piano))
   (obiettivo (sposta C) (sopra-a F)))
```