La KB in clausole di Horn

C1) Negozio(x) ∧ Vende(x, y, z) ∧ Alcolico(y) ∧ Minorenne(z) ⇒ Reo(x)

C2) Possiede(Marco, B)

C3) Birra(B)

C4) Possiede(Marco, x) ∧ Birra(x) ⇒ Vende(SottoCasa, x, Marco)

C5) Birra(x) ⇒ Alcolico(x)

C6) Minimarket(SottoCasa)

C7) Minimarket(x) ⇒ Negozio(x)

C8) Minorenne(Marco)

NOTA: questa KB è costituita da **clausole di Horn del prim'ordine**. In particolare si tratta di clausole che non fanno uso di funzioni. Questa particolare classe di KB è detta **DATALOG**

La KB in clausole di Horn

- È l'unica modellazione possibile?
- **Probabilmente no**
- In generale dato un problema di rappresentazione esisteranno diverse alternative
- George Box (statistico):
  "Now it would be very remarkable if any system existing in the real world could be exactly represented by any simple model. However, cunningly chosen parsimonious models often do provide remarkably useful approximations. For example, the law PV = RT relating pressure P, volume V and temperature T of an "ideal" gas via a constant R is not exactly true for any real gas, but it frequently provides a useful approximation and furthermore its structure is informative since it springs from a physical view of the behavior of gas molecules. For such a model there is no need to ask the question "Is the model true?". If "truth" is to be the "whole truth" the answer must be "No". The only question of interest is "Is the model illuminating and useful?"

  Riassunto talvolta in: **tutti i modelli sono "sbagliati", alcuni sono utili**

Forward Chaining

- L'algoritmo è simile a quello usato nel caso proposizionale
- Le variabili richiedono qualche cautela: un fatto è la **rinomina** di un altro se sono identici tranne nei nomi delle variabili

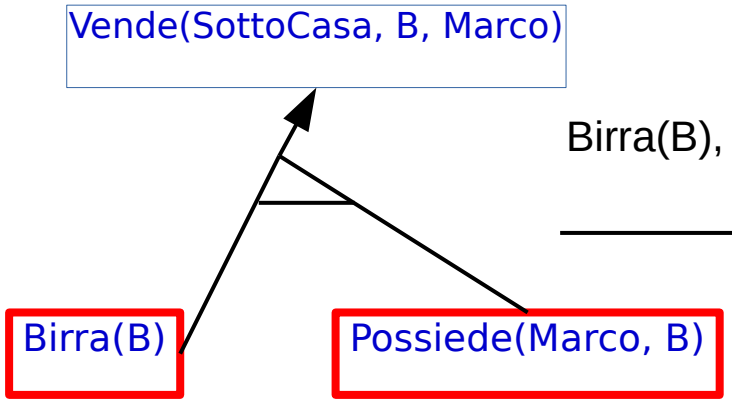
Esempio di rinomina:
Fratello(John, x) e Fratello(John, y)

- **Nel calcolo proposizionale un fatto inferito viene aggiunto alla KB solo se non vi compare già**
- **In FOL bisogna fare un test più complesso: un fatto è aggiunto alla KB solo se non è una rinomina di uno già presente**

Forward chaining

C2) Possiede(Marco, B)
C3) Birra(B)
C4) **Possiede(Marco, x) ∧ Birra(x)** ⇒ Vende(SottoCasa, x, Marco)

Si applica il modus ponens generalizzato



Birra(B), Possiede(Marco, B),
Birra(x) ∧ Possiede(Marco, x) ⇒ vende(SottoCasa, x, Marco)

---

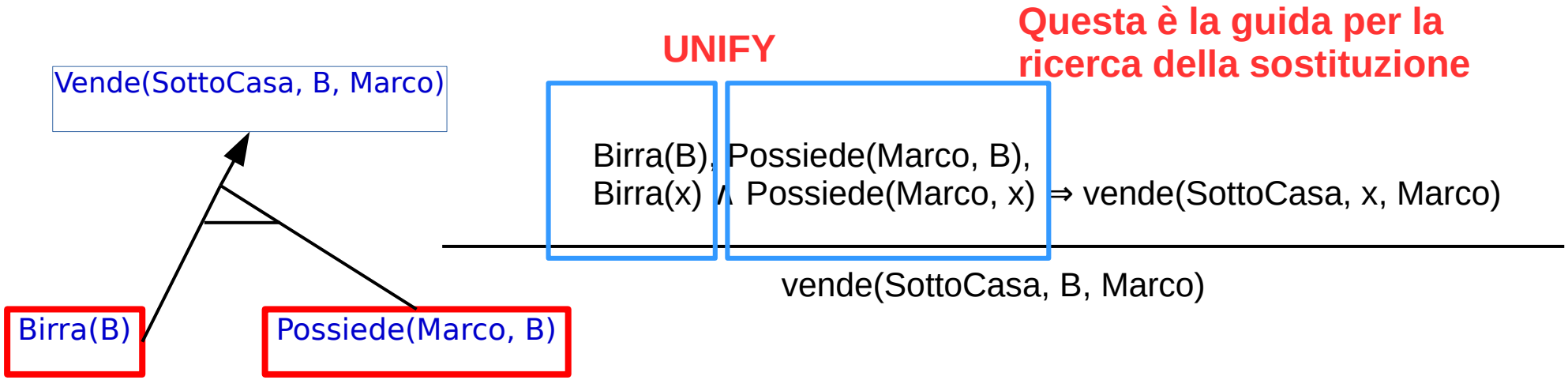
vende(SottoCasa, B, Marco)

Parto da fatti che rendono vero l'antecedente (la premessa) di una implicazione

Forward chaining

C2) Possiede(Marco, B)
C3) Birra(B)
C4) **Possiede(Marco, x) ∧ Birra(x)** ⇒ Vende(SottoCasa, x, Marco)

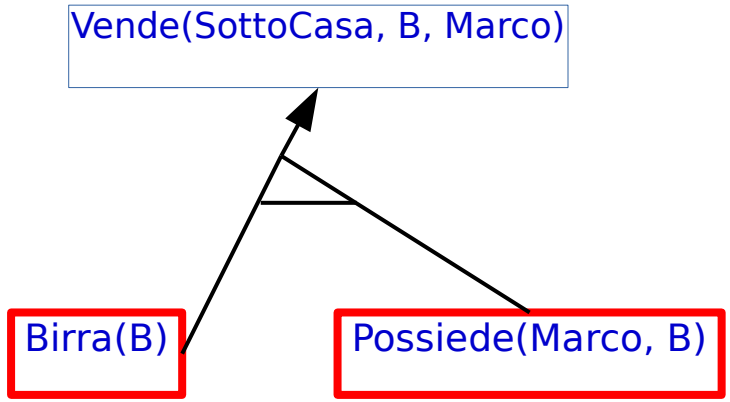
Si applica il modus ponens generalizzato



Parto da fatti che rendono vero l'antecedente (la premessa) di una implicazione e costruisco un grafo AND-OR

Forward chaining

C2) Possiede(Marco, B)
C3) Birra(B)
C4) **Possiede(Marco, x) ∧ Birra(x)** ⇒ Vende(SottoCasa, x, Marco)



Questa inferenza è possibile legando la variabile x a B.

L'inferenza procede creando delle **sostituzioni** per unificazione delle formule:
θ = { x/B }

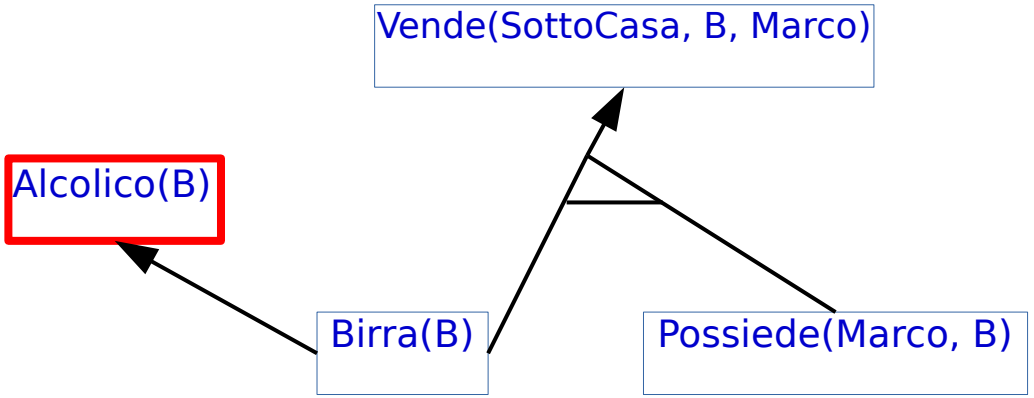
Birra(B)/B = Birra(x)/B
Possiede(Marco,x)/B =
Possiede(Marco, B)/B

Parto da fatti che rendono vero l'antecedente (la premessa) di una implicazione e costruisco un grafo AND-OR

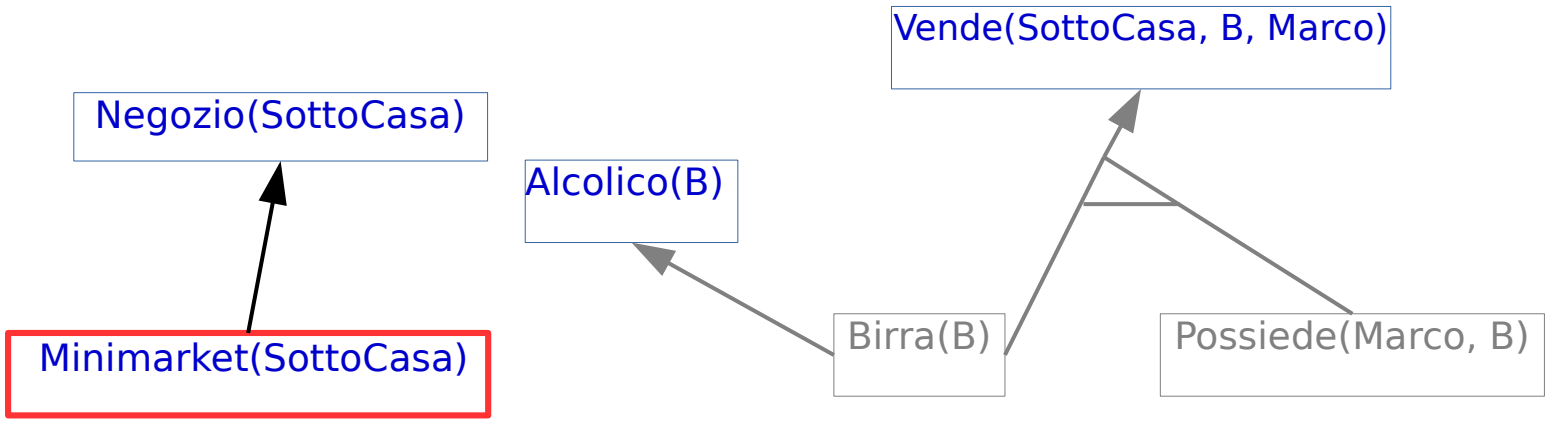
Forward Chaining

C2) Possiede(Marco, B)
C3) Birra(B)
C4) Possiede(Marco, x) ∧ Birra(x) ⇒ Vende(SottoCasa, x, Marco)
C5) Birra(x) ⇒ Alcolico(x)

Non si propagano solo i valori di verità. Si propagano anche le sostituzioni, che fanno da collante fra le diverse applicazioni del MPG

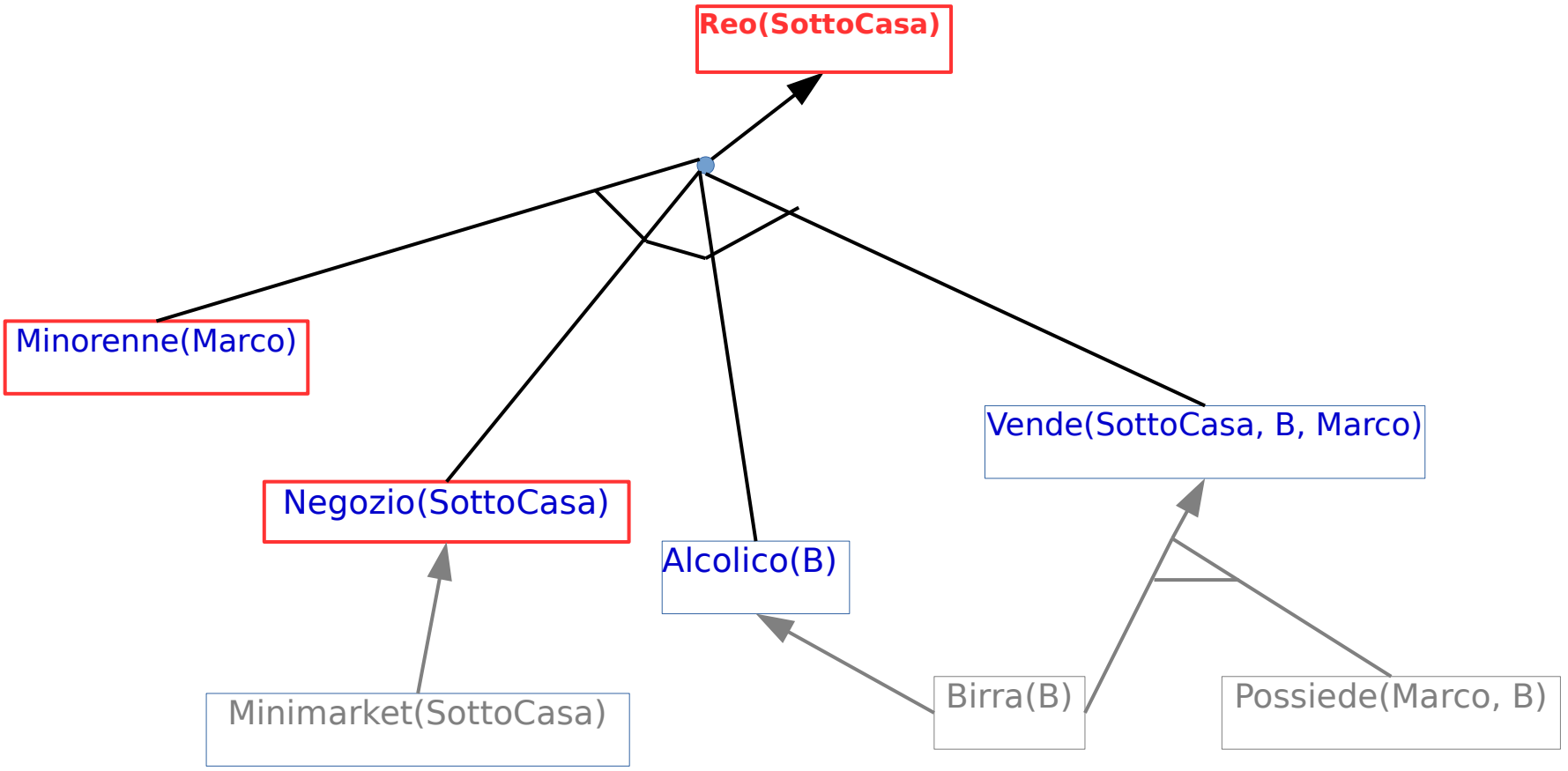


Uso del forward chaining



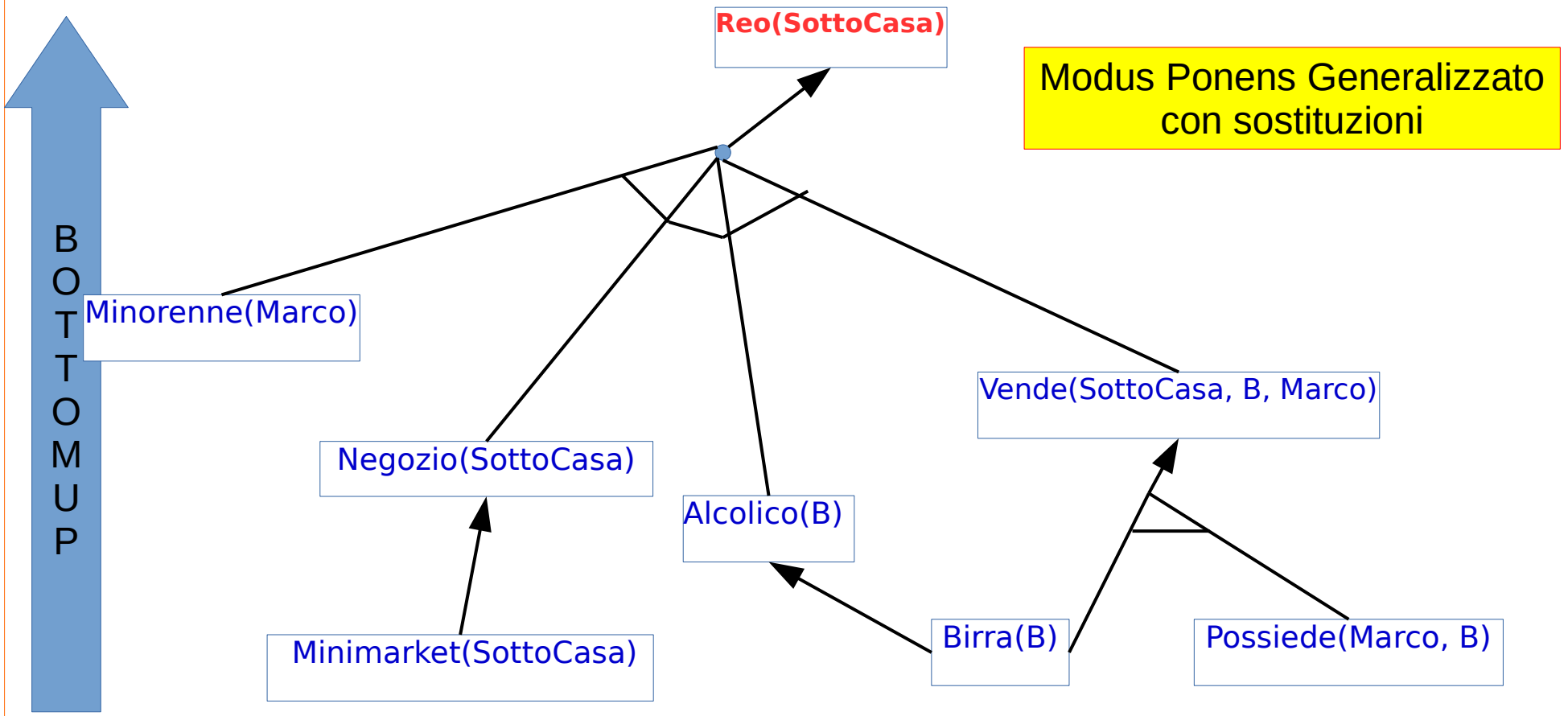
Uso del forward chaining

Inferisco che Sotto Casa è reo della vendita



Uso del forward chaining

Grafo AND-OR completo



Proprietà del forward chaining

- **Correttezza**
  - l'algoritmo è **corretto** perché si basa sulla regola di inferenza corretta GMP
- **Completezza**
  - È **completo** per KB costituite da **clausole di Horn**?
    - Se la KB è **DATALOG**, è **completo e termina**
    - Se la KB **prevede funzioni** – teorema di Herbrand – l'algoritmo **può non terminare** se la risposta non è implicata

Backward Chaining (BC)

- Come nel caso proposizionale il **ragionamento è guidato dall'obiettivo**
- L'obiettivo viene inserito in uno **stack**
- Poi iterativamente si estrae un obiettivo dallo stack e si cercano **le clausole la cui testa è unificabile** con l'obiettivo:
  - Quando un obiettivo unifica con un fatto (clausola con corpo vuoto), viene semplicemente rimosso (è risolto)
  - Altrimenti per ogni clausola che soddisfa quanto sopra si avvia un procedimento ricorsivo che inserisce nella pila le premesse della clausola, in cui le variabili saranno state debitamente sostituite
- Quando la pila è vuota si termina con successo
- Altrimenti se non è possibile applicare altre inferenze si termina con fallimento

Esempio

Stack: Reo(SottoCasa)

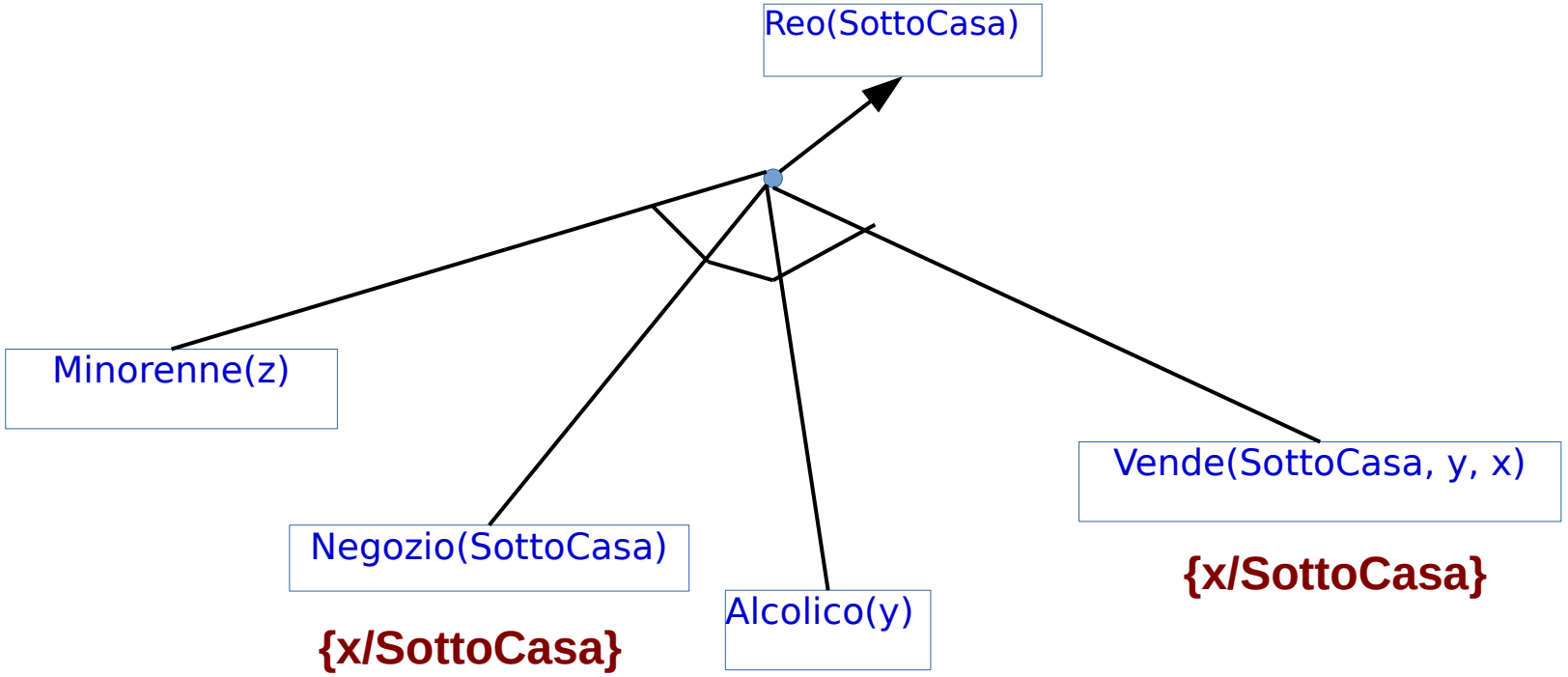
Unifica con la testa della clausola tramite **θ1 = {x/SottoCasa}**:
C1) Negozio(x) ∧ Vende(x, y, z) ∧ Alcolico(y) ∧ Minorenne(z) ⇒ Reo(x)

Si applica la sostituzione al corpo di C1, tutti i congiunti diventano obiettivi da dimostrare e vengono inseriti nello stack:
Stack: Negozio(SottoCasa), Vende(SottoCasa, y, z), Alcolico(y), Minorenne(z)

Esempio

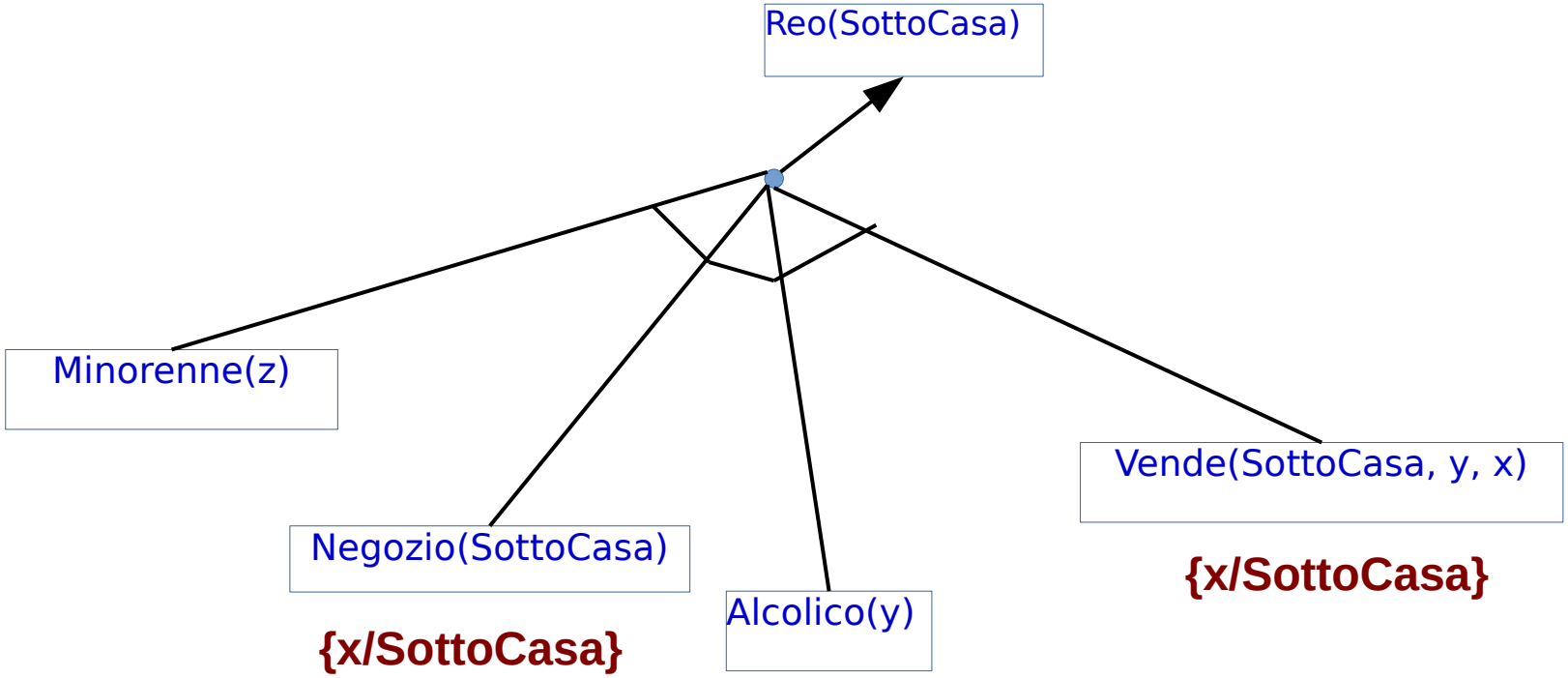
C1) Negozio(x) ∧ Vende(x, y, z) ∧ Alcolico(y) ∧ Minorenne(z) ⇒ Reo(x)

Stack: Negozio(SottoCasa), Vende(SottoCasa, y, z), Alcolico(y), Minorenne(z)



Esempio

C1) Negozio(x) ∧ Vende(x, y, z) ∧ Alcolico(y) ∧ Minorenne(z) ⇒ Reo(x)
Stack: Vende(SottoCasa, y, z), Alcolico(y), Minorenne(z)
Risolviamo ricorsivamente Negozio(SottoCasa)

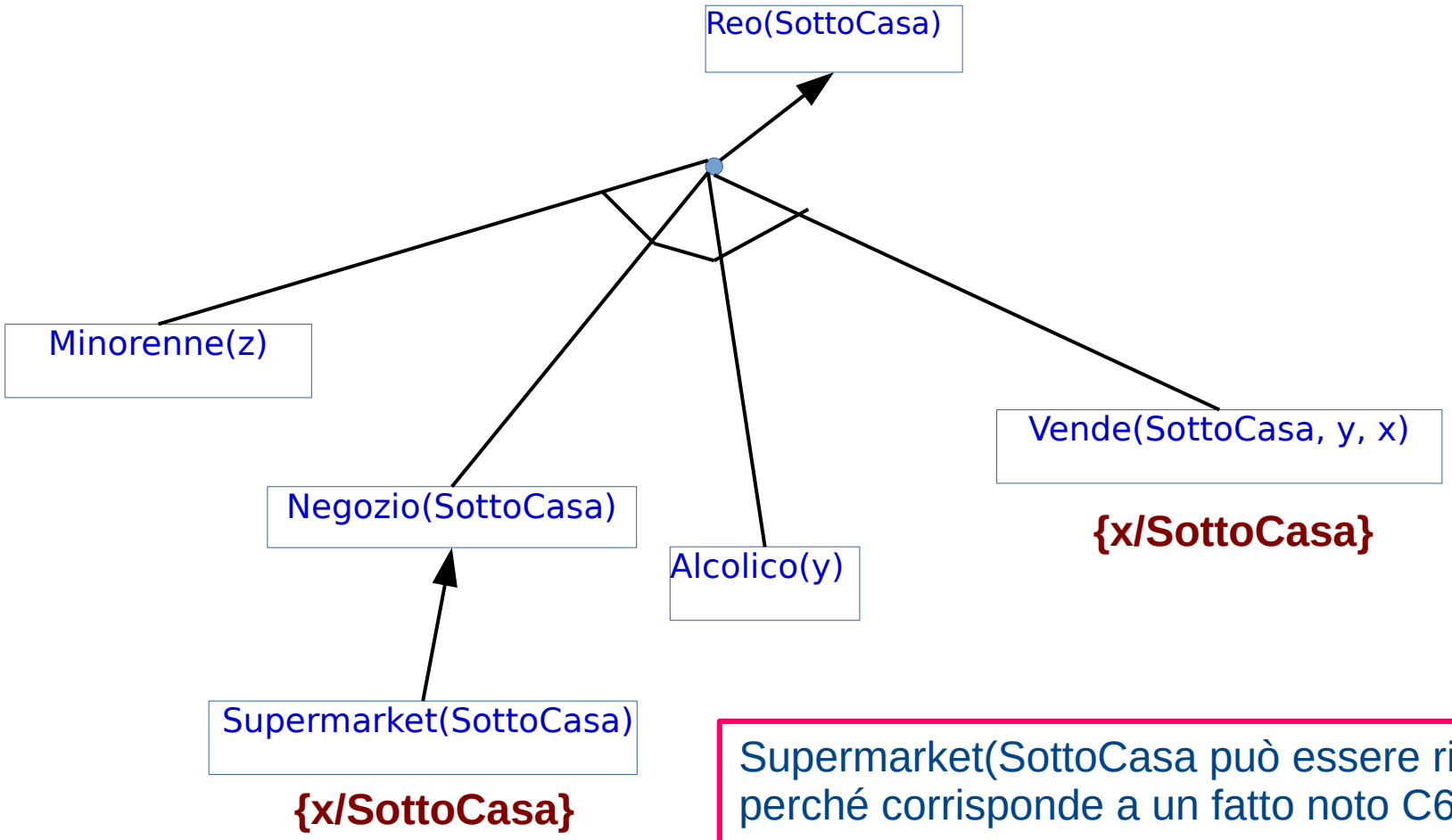


unifica con la testa della clausola C7) Supermarket(x) ⇒ Negozio(x)

Esempio

C1) Negozio(x) ∧ Vende(x, y, z) ∧ Alcolico(y) ∧ Minorenne(z) ⇒ Reo(x)

Stack: ~~**Supermarket(SottoCasa)**~~, Vende(SottoCasa, y, z), Alcolico(y), Minorenne(z)

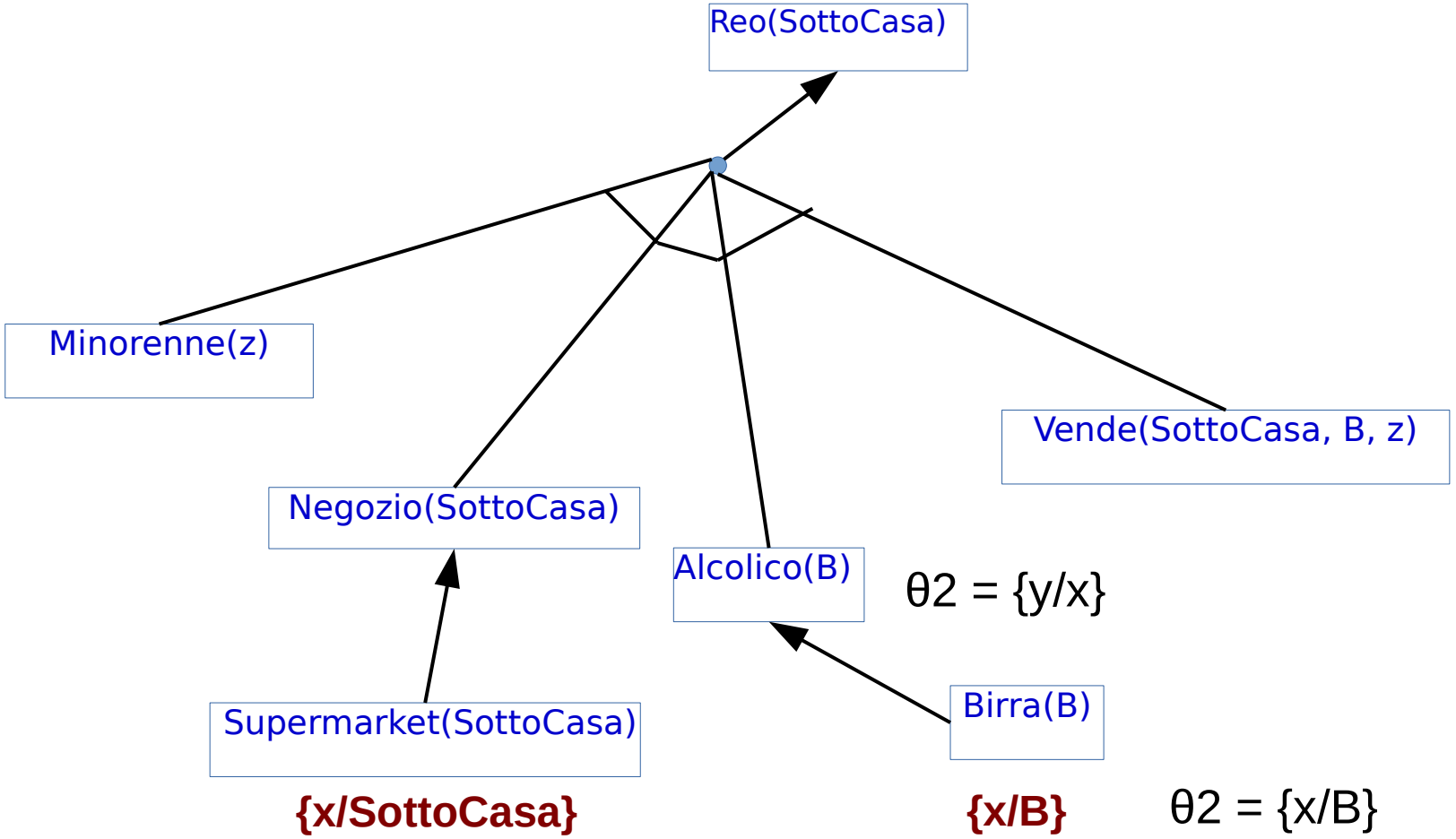


Supermarket(SottoCasa può essere rimosso perché corrisponde a un fatto noto C6)

Esempio

C3) Birra(B)

C5) Birra(x) ⇒ Alcolico(x)



Composizione delle sostituzioni

- l'algoritmo BC applica la composizione delle sostituzioni:

  COMPOSE(θ1, θ2) = θ3

- Vale la proprietà:

  SUBST(**COMPOSE(θ1, θ2)**, F) = SUBST(θ2, SUBST(θ1, F))

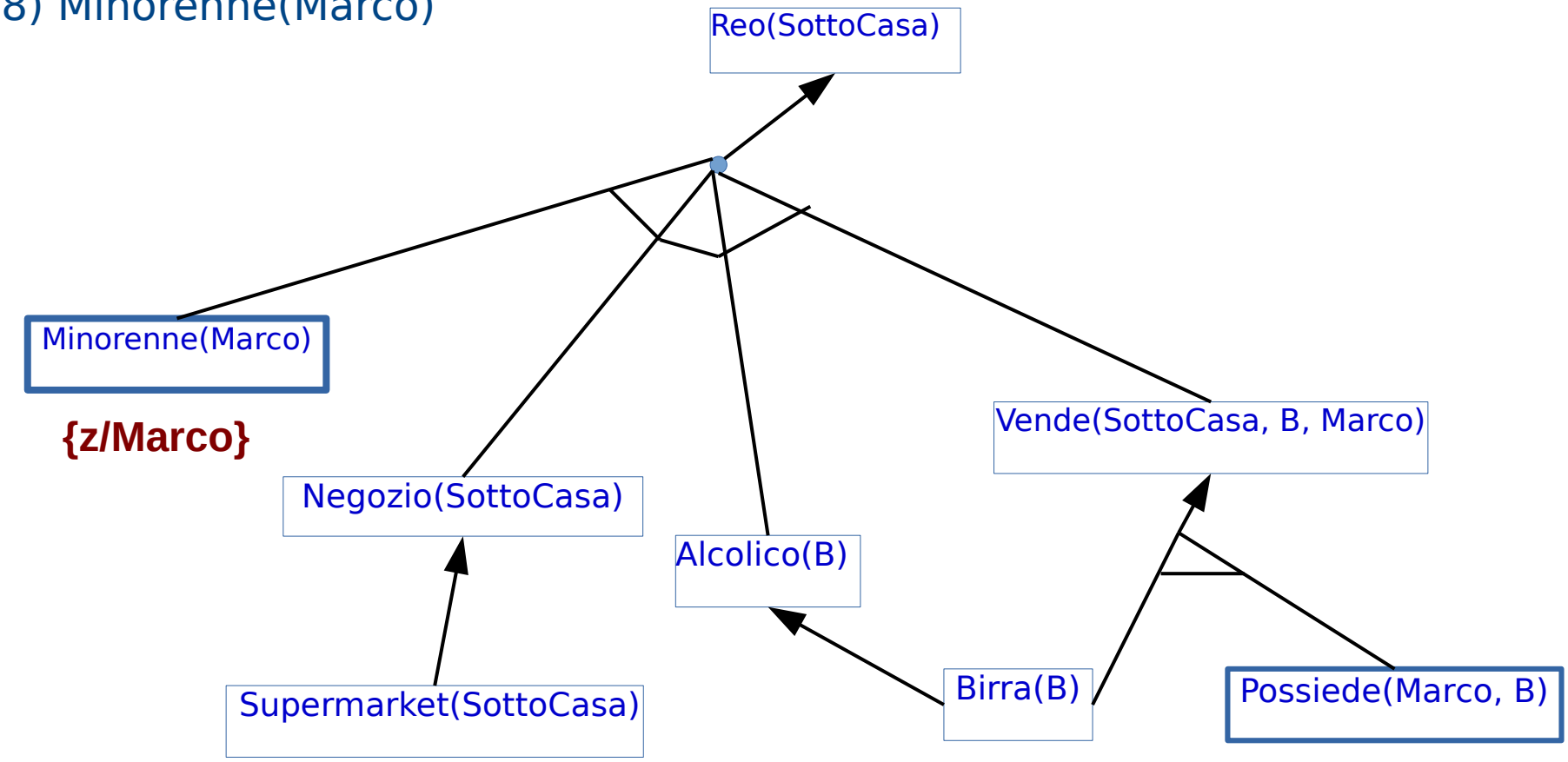
- Nell'esempio abbiamo:
  Negozio(x) ∧ Vende(x, y, z) ∧ Alcolico(y) ∧ Minorenne(z) ⇒ Reo(x)
  Birra(x) ⇒ Alcolico(x)
  Birra(B)
- Occorre concatenare le sostituzioni per effettuare l'inferenza corretta

Esempio

C2) Possiede(Marco, B)

C4) Possiede(Marco, x) ∧ Birra(x) ⇒ Vende(SottoCasa, x, Marco)

C8) Minorenne(Marco)



Valutazionde del backward chaining

- **Corretto**
- **Incompleto** in quanto implementa una *strategia depth-first*, può incorrere in loop infiniti e generare stati ripetuti
- Come la versione per il calcolo proposizionale è più efficiente del forward chaining