IKI 30320: Sistem Cerdas
Kuliah 13: Inference in FOL

Ruli Manurung

Fakultas Ilmu Komputer
Universitas Indonesia

5 November 2007

Outline

Outline

Ide tahun 60-an

- Kita sudah melihat mekanisme *inference* untuk *propositional logic*
  - Inference rule: Modus Ponens
    Normal form: Horn clause
    Algoritma: Forward chaining, Backward chaining
  - Inference rule: Resolution
    Normal form: CNF
    Algoritma: Proof-by-contradiction
- Pendekatan-pendekatan ini *sound* dan *complete*.

Ide tahun 60-an

- Kita sudah melihat mekanisme *inference* untuk *propositional logic*
  - Inference rule: Modus Ponens
    Normal form: Horn clause
    Algoritma: Forward chaining, Backward chaining
  - Inference rule: Resolution
    Normal form: CNF
    Algoritma: Proof-by-contradiction
- Pendekatan-pendekatan ini *sound* dan *complete*.

Cara mudah melakukan inference FOL:

Jika *KB* dan *query* dalam FOL bisa diterjemahkan ke dalam PL, beres!

Instantiation

- Ground term: sebuah *term* tanpa variable, mis: *Ani, Ayah(Anto), 2007*
- Instantiation: kalimat di mana sebuah variable diganti dengan sebuah ground term (diperoleh dengan mengaplikasikan sebuah *substitution*)

Contoh

$\alpha = \forall\, x \;\; mahasiswa(x, FasilkomUI) \Rightarrow pintar(x)$

$\beta = \exists\, x \;\; mahasiswa(x, Gundar) \wedge pintar(x)$

$\sigma = \{x/Anto\}$

SUBST(σ, α) menghasilkan *instantiation*:

Instantiation

- Ground term: sebuah *term* tanpa variable, mis: *Ani, Ayah(Anto), 2007*
- Instantiation: kalimat di mana sebuah variable diganti dengan sebuah ground term (diperoleh dengan mengaplikasikan sebuah *substitution*)

Contoh

$\alpha = \forall x \;\; mahasiswa(x, FasilkomUI) \Rightarrow pintar(x)$
$\beta = \exists x \;\; mahasiswa(x, Gundar) \wedge pintar(x)$
$\sigma = \{x/Anto\}$
$\textsc{Subst}(\sigma, \alpha)$ menghasilkan *instantiation*:
$mahasiswa(Anto, FasilkomUI) \Rightarrow pintar(Anto)$
$\textsc{Subst}(\sigma, \beta)$ menghasilkan *instantiation*:

Instantiation

- Ground term: sebuah *term* tanpa variable, mis: *Ani, Ayah(Anto), 2007*
- Instantiation: kalimat di mana sebuah variable diganti dengan sebuah ground term (diperoleh dengan mengaplikasikan sebuah *substitution*)

Contoh

$\alpha = \forall\, x \;\; mahasiswa(x, FasilkomUI) \Rightarrow pintar(x)$

$\beta = \exists\, x \;\; mahasiswa(x, Gundar) \wedge pintar(x)$

$\sigma = \{x/Anto\}$

$\textsc{Subst}(\sigma, \alpha)$ menghasilkan *instantiation*:

$mahasiswa(Anto, FasilkomUI) \Rightarrow pintar(Anto)$

$\textsc{Subst}(\sigma, \beta)$ menghasilkan *instantiation*:

$mahasiswa(Anto, Gundar) \wedge pintar(Anto)$

Universal Instantiation

Sebuah kalimat dengan universal quantifier (∀) meng-*entail* semua *instantiation*-nya:

$$\frac{\forall\, v \;\; \alpha}{SUBST(\{v/g\}, \alpha)}$$

untuk sembarang variable v dan ground term g

Contoh

$\forall\, x \;\; King(x) \land Greedy(x) \Rightarrow Evil(x)$ meng-entail:

$$\begin{array}{l} King(John) \land Greedy(John) \Rightarrow Evil(John) \\ King(Richard) \land Greedy(Richard) \Rightarrow Evil(Richard) \\ King(Father(John)) \land Greedy(Father(John)) \Rightarrow Evil(Father(John)) \\ \vdots \end{array}$$

Existential Instantiation

Untuk sembarang variable v, kalimat α dan constant k yang tidak muncul di knowledge-base:

$$\frac{\exists\, v \quad \alpha}{SUBST(\{v/k\}, \alpha)}$$

Contoh

$\exists\, x \quad Crown(x) \land OnHead(x, John)$ meng-entail:
$Crown(C_1) \land OnHead(C_1, John)$
dengan syarat C_1 adalah *constant symbol* yang baru, disebut *Skolem constant*

Menghilangkan quantifier dan variable

Menghilangkan $\forall$

- Universal instantiation bisa digunakan berkali-kali untuk menambahkan kalimat baru.
- KB yang baru *logically equivalent* dengan yang lama.

Menghilangkan $\exists$

- Existential instantiation cukup digunakan sekali untuk menggantikan kalimat existential.
- KB yang baru tidak *logically equivalent* dengan yang lama, tetapi *satisfiable* jhj KB yang lama juga *satisfiable* $\rightarrow$ inferentially equivalent

Contoh

Andaikan *KB* berisi kalimat-kalimat berikut:

$\forall\, x \quad King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
$\forall\, y \quad Greedy(y)$
$King(John)$
$Brother(Richard, John)$

Jika kita mengambil semua kemungkinan instantiation dari kalimat universal, kita dapatkan KB sbb:

$King(John) \wedge Greedy(John) \Rightarrow Evil(John)$
$King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$
$Greedy(John)$
$Greedy(Richard)$
$King(John)$
$Brother(Richard, John)$

KB yang baru dikatakan propositionalized: proposition symbol-nya: $King(John)$, $Greedy(John)$, $Evil(John)$, $King(Richard)$ etc.

Inference FOL menggunakan inference PL

- Ide dasar: ubah *KB* + *query* dari FOL menjadi PL, lalu gunakan *resolution*.

Inference FOL menggunakan inference PL

- Ide dasar: ubah *KB* + *query* dari FOL menjadi PL, lalu gunakan *resolution*.

Masalah: dengan adanya function, jumlah ground term menjadi infinite

$Greedy(Father(John))$
$Greedy(Father(Father(John)))$
$Greedy(Father(Father(Father(John))))$, dst.

Inference FOL menggunakan inference PL

- Ide dasar: ubah *KB* + *query* dari FOL menjadi PL, lalu gunakan *resolution*.

Masalah: dengan adanya function, jumlah ground term menjadi infinite

$Greedy(Father(John))$
$Greedy(Father(Father(John)))$
$Greedy(Father(Father(Father(John))))$, dst.

- Teorema Herbrand (1930): jika FOL KB $\models \alpha$, ada sebuah finite subset PL KB $\models \alpha$.
- Ide dasar: For $n = 0$ to ∞
  Buat propositional KB_n dengan depth-n ground term
  Periksa apakah $KB_n \models \alpha$

Inference FOL menggunakan inference PL

- Ide dasar: ubah *KB* + *query* dari FOL menjadi PL, lalu gunakan *resolution*.

Masalah: dengan adanya function, jumlah ground term menjadi infinite

$Greedy(Father(John))$
$Greedy(Father(Father(John)))$
$Greedy(Father(Father(Father(John))))$, dst.

- Teorema Herbrand (1930): jika FOL KB $\models \alpha$, ada sebuah finite subset PL KB $\models \alpha$.
- Ide dasar: For $n = 0$ to ∞
  Buat propositional KB_n dengan depth-n ground term
  Periksa apakah $KB_n \models \alpha$

Masalah (lagi!)

kalau α di-*entail* OK, kalau tidak $\rightarrow$ infinite loop.

Inference FOL menggunakan inference PL

- Ide dasar: ubah *KB* + *query* dari FOL menjadi PL, lalu gunakan *resolution*.

Masalah: dengan adanya function, jumlah ground term menjadi infinite

$Greedy(Father(John))$
$Greedy(Father(Father(John)))$
$Greedy(Father(Father(Father(John))))$, dst.

- Teorema Herbrand (1930): jika FOL KB $\models \alpha$, ada sebuah finite subset PL KB $\models \alpha$.
- Ide dasar: For $n = 0$ to ∞
  Buat propositional KB_n dengan depth-n ground term
  Periksa apakah $KB_n \models \alpha$

Masalah (lagi!)

kalau α di-*entail* OK, kalau tidak $\rightarrow$ infinite loop.

- Teorema Church-Turing (1936): *Entailment* untuk FOL bersifat semidecidable.

Masalah dengan *propositionalization*

- Propositionalization menghasilkan banyak kalimat irelevan.
- Contohnya, dari KB berikut:

  $$\forall x \;\; King(x) \wedge Greedy(x) \Rightarrow Evil(x)$$
  $$\forall y \;\; Greedy(y)$$
  $$King(John)$$
  $$Brother(Richard, John)$$

  manusia bisa cepat mengerti kalau $Evil(John)$, namun propositionalization menghasilkan:

  $$King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$$
  $$Greedy(Richard)$$

  yang tidak relevan
- Dengan p buah predicate k-ary dan n constant, ada $p \times n^k$ instantiation!

Outline

1. Mengubah FOL ke PL
2. Unification
3. Inference Rule untuk FOL
4. Forward chaining
5. Backward chaining
6. Logic programming
7. Resolution
8. Ringkasan

Unification

Isi KB

$\forall x \;\; King(x) \land Greedy(x) \Rightarrow Evil(x)$
$\forall y \;\; Greedy(y)$
$King(John)$
$Brother(Richard, John)$

Inference bahwa $KB \models Evil(John)$ bisa langsung disimpulkan jika kita bisa mencari *substitution* θ sehingga $King(x)$ dan $Greedy(x)$ bisa "dicocokkan" dengan $King(John)$ dan $Greedy(y)$.
Contoh: $\theta = \{x/John, y/John\}$

Definisi unification

$\textsc{Unify}(\alpha, \beta) = \theta$ jika $\textsc{Subst}(\alpha,\theta) = \textsc{Subst}(\beta,\theta)$

Contah unification

p	q	θ
$Sayang(Anto, x)$	$Sayang(Anto, Ani)$	

Contah unification

p	q	θ
$Sayang(Anto, x)$	$Sayang(Anto, Ani)$	$\{x/Ani\}$
$Sayang(Anto, x)$	$Sayang(y, Ani)$	

Contah unification

p	q	θ
$Sayang(Anto, x)$	$Sayang(Anto, Ani)$	$\{x/Ani\}$
$Sayang(Anto, x)$	$Sayang(y, Ani)$	$\{x/Ani, y/Anto\}$
$Sayang(Anto, x)$	$Sayang(y, Ibu(y))$	

Contah unification

p	q	θ
$Sayang(Anto, x)$	$Sayang(Anto, Ani)$	$\{x/Ani\}$
$Sayang(Anto, x)$	$Sayang(y, Ani)$	$\{x/Ani, y/Anto\}$
$Sayang(Anto, x)$	$Sayang(y, Ibu(y))$	$\{y/Anto, x/Ibu(Anto)$
$Sayang(Anto, x)$	$Sayang(x, Ani)$	

Contah unification

p	q	θ
$Sayang(Anto, x)$	$Sayang(Anto, Ani)$	$\{x/Ani\}$
$Sayang(Anto, x)$	$Sayang(y, Ani)$	$\{x/Ani, y/Anto\}$
$Sayang(Anto, x)$	$Sayang(y, Ibu(y))$	$\{y/Anto, x/Ibu(Anto)$
$Sayang(Anto, x)$	$Sayang(x, Ani)$	*fail*

- Standardized apart variable menghilangkan *overlap*, mis: $Sayang(x_{101}, Ani)$
- Lihat Figure 9.1 R&N2e untuk implementasi algoritma Unification

Outline

Generalized Modus Ponens

Inference rule GMP

$$\frac{p_1{}',\ \ p_2{}',\ \ldots,\ p_n{}',\ \ (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{q\theta}$$

di mana $p_i{}'\theta = p_i\theta$ untuk semua i

$$\begin{array}{ll} p_1{}' = King(John) & p_1 = King(x) \\ p_2{}' = Greedy(y) & p_2 = Greedy(x) \\ \theta = \{x/John, y/John\} & q = Evil(x) \\ q\theta = Evil(John) & \end{array}$$

- GMP dengan KB yang berisi definite clauses (seperti Horn clause pada PL): $p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q$
- Semua variable diasumsikan universally quantified
- GMP adalah hasil lifting MP: “mengangkat” inference rule PL ke FOL. Ada versi *lifted* untuk forward & backward chaining, resolution
- Kelebihan dibanding *propositionalization*: hanya melakukan *substitution* yang dibutuhkan oleh *inference*

Contoh knowledge base

Versi bahasa Inggris

"The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American."

Buktikan bahwa Col. West adalah criminal!

Penerjemahan Bhs Inggris ke FOL

- . . . it is a crime for an American to sell weapons to hostile nations:

Penerjemahan Bhs Inggris ke FOL

- . . . it is a crime for an American to sell weapons to hostile nations:
  $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$

Penerjemahan Bhs Inggris ke FOL

- . . . it is a crime for an American to sell weapons to hostile nations:
  $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$
- Nono . . . has some missiles:

Penerjemahan Bhs Inggris ke FOL

- . . . it is a crime for an American to sell weapons to hostile nations:
  $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$
- Nono . . . has some missiles:
  $\exists\, x\ Owns(Nono, x) \wedge Missile(x)$

Penerjemahan Bhs Inggris ke FOL

- . . . it is a crime for an American to sell weapons to hostile nations:
  $American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \Rightarrow Criminal(x)$
- Nono . . . has some missiles:
  $\exists\, x\ Owns(Nono, x) \land Missile(x)$
  $Owns(Nono, M_1)$ and $Missile(M_1)$ (Skolemization)

Penerjemahan Bhs Inggris ke FOL

- . . . it is a crime for an American to sell weapons to hostile nations:
  $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$
- Nono . . . has some missiles:
  $\exists\, x\ Owns(Nono, x) \wedge Missile(x)$
  $Owns(Nono, M_1)$ and $Missile(M_1)$ (Skolemization)
- . . . all of its missiles were sold to it by Colonel West

Penerjemahan Bhs Inggris ke FOL

- . . . it is a crime for an American to sell weapons to hostile nations:
  $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$
- Nono . . . has some missiles:
  $\exists\, x\ Owns(Nono, x) \wedge Missile(x)$
  $Owns(Nono, M_1)$ and $Missile(M_1)$ (Skolemization)
- . . . all of its missiles were sold to it by Colonel West
  $Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$

Penerjemahan Bhs Inggris ke FOL

- . . . it is a crime for an American to sell weapons to hostile nations:
  $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$
- Nono . . . has some missiles:
  $\exists\, x\ Owns(Nono, x) \wedge Missile(x)$
  $Owns(Nono, M_1)$ and $Missile(M_1)$ (Skolemization)
- . . . all of its missiles were sold to it by Colonel West
  $Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$
- Missiles are weapons:

Penerjemahan Bhs Inggris ke FOL

- . . . it is a crime for an American to sell weapons to hostile nations:
  $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$
- Nono . . . has some missiles:
  $\exists\, x\ Owns(Nono, x) \wedge Missile(x)$
  $Owns(Nono, M_1)$ and $Missile(M_1)$ (Skolemization)
- . . . all of its missiles were sold to it by Colonel West
  $Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$
- Missiles are weapons:
  $Missile(x) \Rightarrow Weapon(x)$

Penerjemahan Bhs Inggris ke FOL

- . . . it is a crime for an American to sell weapons to hostile nations:
  $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$
- Nono . . . has some missiles:
  $\exists\, x\ Owns(Nono, x) \wedge Missile(x)$
  $Owns(Nono, M_1)$ and $Missile(M_1)$ (Skolemization)
- . . . all of its missiles were sold to it by Colonel West
  $Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$
- Missiles are weapons:
  $Missile(x) \Rightarrow Weapon(x)$
- An enemy of America counts as “hostile”:

Penerjemahan Bhs Inggris ke FOL

- . . . it is a crime for an American to sell weapons to hostile nations:
  $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$
- Nono . . . has some missiles:
  $\exists\, x\ Owns(Nono, x) \wedge Missile(x)$
  $Owns(Nono, M_1)$ and $Missile(M_1)$ (Skolemization)
- . . . all of its missiles were sold to it by Colonel West
  $Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$
- Missiles are weapons:
  $Missile(x) \Rightarrow Weapon(x)$
- An enemy of America counts as “hostile”:
  $Enemy(x, America) \Rightarrow Hostile(x)$

Penerjemahan Bhs Inggris ke FOL

- . . . it is a crime for an American to sell weapons to hostile nations:
  $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$
- Nono . . . has some missiles:
  $\exists\, x\ Owns(Nono, x) \wedge Missile(x)$
  $Owns(Nono, M_1)$ and $Missile(M_1)$ (Skolemization)
- . . . all of its missiles were sold to it by Colonel West
  $Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$
- Missiles are weapons:
  $Missile(x) \Rightarrow Weapon(x)$
- An enemy of America counts as “hostile”:
  $Enemy(x, America) \Rightarrow Hostile(x)$
- West, who is American . . .

Penerjemahan Bhs Inggris ke FOL

- . . . it is a crime for an American to sell weapons to hostile nations:
  $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$
- Nono . . . has some missiles:
  $\exists\, x\ Owns(Nono, x) \wedge Missile(x)$
  $Owns(Nono, M_1)$ and $Missile(M_1)$ (Skolemization)
- . . . all of its missiles were sold to it by Colonel West
  $Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$
- Missiles are weapons:
  $Missile(x) \Rightarrow Weapon(x)$
- An enemy of America counts as “hostile”:
  $Enemy(x, America) \Rightarrow Hostile(x)$
- West, who is American . . .
  $American(West)$

Penerjemahan Bhs Inggris ke FOL

- . . . it is a crime for an American to sell weapons to hostile nations:
  $American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \Rightarrow Criminal(x)$
- Nono . . . has some missiles:
  $\exists\, x\ Owns(Nono, x) \land Missile(x)$
  $Owns(Nono, M_1)$ and $Missile(M_1)$ (Skolemization)
- . . . all of its missiles were sold to it by Colonel West
  $Missile(x) \land Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$
- Missiles are weapons:
  $Missile(x) \Rightarrow Weapon(x)$
- An enemy of America counts as “hostile”:
  $Enemy(x, America) \Rightarrow Hostile(x)$
- West, who is American . . .
  $American(West)$
- The country Nono, an enemy of America . . .

Penerjemahan Bhs Inggris ke FOL

- . . . it is a crime for an American to sell weapons to hostile nations:
  $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$
- Nono . . . has some missiles:
  $\exists\, x\ Owns(Nono, x) \wedge Missile(x)$
  $Owns(Nono, M_1)$ and $Missile(M_1)$ (Skolemization)
- . . . all of its missiles were sold to it by Colonel West
  $Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$
- Missiles are weapons:
  $Missile(x) \Rightarrow Weapon(x)$
- An enemy of America counts as “hostile”:
  $Enemy(x, America) \Rightarrow Hostile(x)$
- West, who is American . . .
  $American(West)$
- The country Nono, an enemy of America . . .
  $Enemy(Nono, America)$

Perhatikan:

Semua kalimat KB ini berbentuk definite clause.

Outline

Forward chaining pada FOL dengan GMP

- Mirip dengan forward chaining pada PL
- Mulai dari fakta yang diketahui (clause tanpa premise), mis: $Owns(Nono, M_1), Missile(M_1)$
- "Aktifkan" (*trigger*) rule yang premise-nya diketahui (satisfied) $\longrightarrow$ tambahkan kesimpulan rule ke KB, mis: $Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$
- Ulangi sampai *query* terbukti, atau tidak ada fakta baru yang bisa ditambahkan ke KB.
- "Cocokkan" premise-premise setiap rule dengan fakta yang diketahui $\longrightarrow$ pattern-matching dengan *unification*

Forward chaining pada FOL dengan GMP

Algoritma forward chaining

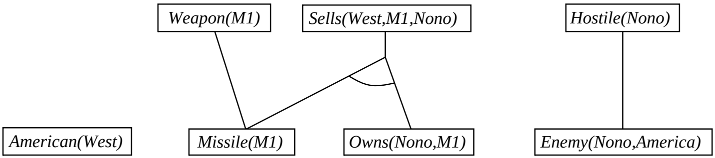
function FOL-FC-ASK(KB, α) **returns** a substitution or *false*

  repeat until *new* is empty
    new $\leftarrow \{\,\}$
    for each sentence *r* **in** *KB* **do**
      $(p_1 \wedge \ldots \wedge p_n \implies q) \leftarrow$ STANDARDIZE-APART(r)
      for each θ such that $(p_1 \wedge \ldots \wedge p_n)\theta = (p'_1 \wedge \ldots \wedge p'_n)\theta$
            for some $p'_1, \ldots, p'_n$ in *KB*
        $q' \leftarrow$ SUBST(θ, q)
        if q' isn't a renaming of sentence in *KB* or *new* **then do**
          add q' to *new*
          $\phi \leftarrow$ UNIFY(q', α)
          if ϕ is not *fail* **then return** ϕ
    add *new* to *KB*
  return *false*

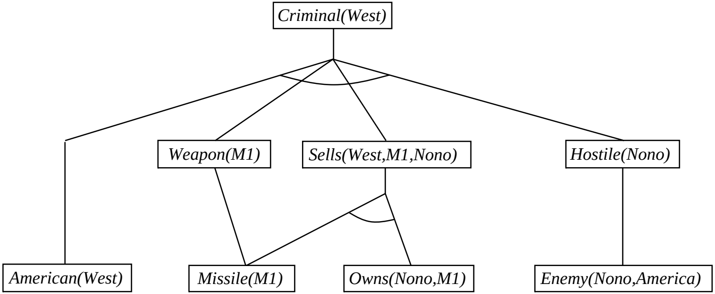
Contoh Forward Chaining FOL

$\boxed{American(West)}$ $\boxed{Missile(M1)}$ $\boxed{Owns(Nono,M1)}$ $\boxed{Enemy(Nono,America)}$

Contoh Forward Chaining FOL



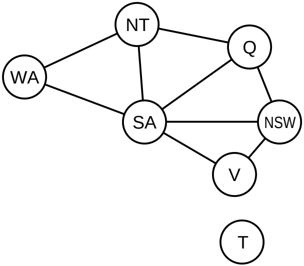
Contoh Forward Chaining FOL



Sifat Forward Chaining

- *Sound* dan *complete* untuk first-order definite clause.
- Datalog = first-order definite clause tanpa function.
  Time complexity FC pada Datalog $\rightarrow$ polynomial
- Tapi pada kasus umum, bisa infinite loop kalau α tidak di-entail. (Konsekuensi dari teorema Church-Turing: entailment adalah *semidecidable*)
- Proses pattern matching pada premise NP-hard.

Pattern matching premise NP-hard?



$$\begin{array}{l} Diff(wa, nt) \wedge Diff(wa, sa) \wedge \\ Diff(nt, q) \wedge Diff(nt, sa) \wedge \\ Diff(q, nsw) \wedge Diff(q, sa) \wedge \\ Diff(nsw, v) \wedge Diff(nsw, sa) \wedge \\ Diff(v, sa) \Rightarrow Colorable() \end{array}$$

$$\begin{array}{l} Diff(Red, Blue) \\ Diff(Red, Green) \\ Diff(Green, Red) \\ Diff(Green, Blue) \\ Diff(Blue, Red) \\ Diff(Blue, Green) \end{array}$$

- Query ASK(*KB*,*Colorable*()) jhj CSP-nya menemui solusi!
- Terdapat kasus CSP 3SAT (satisfiability pada CNF dengan clause berukuran 3 literal) yang diketahui NP-hard.

Outline

Backward chaining pada FOL dengan GMP

Algoritma backward chaining

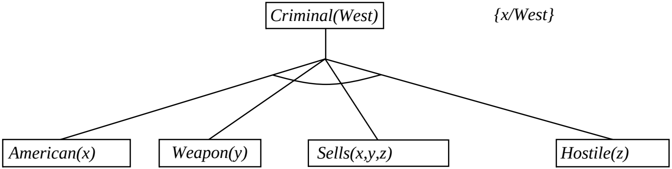
function FOL-BC-ASK(*KB*, goals, θ) **returns** a set of substitutions
  inputs: *KB*, a knowledge base
    goals, a list of conjuncts forming a query
    θ, the current substitution, initially the empty substitution $\{\,\}$
  local variables: *ans*, a set of substitutions, initially empty

  if *goals* is empty **then return** $\{\theta\}$
  $q' \leftarrow$ SUBST(θ, FIRST(*goals*))
  for each *r* **in** *KB* where STANDARDIZE-APART(*r*) = ($p_1 \wedge \ldots \wedge p_n \Rightarrow q$)
    and $\theta' \leftarrow$ UNIFY(q, q') succeeds
      ans $\leftarrow$ FOL-BC-ASK(*KB*, $[p_1, \ldots, p_n|$REST(*goals*)$]$, COMPOSE(θ', θ)) $\cup$ *ans*
  return *ans*

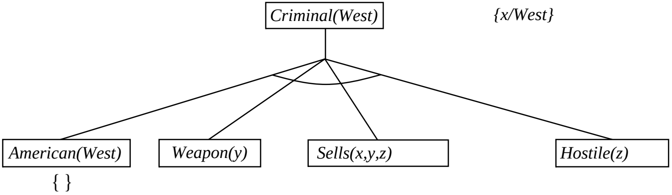
Contoh Backward Chaining FOL

$\boxed{Criminal(West)}$

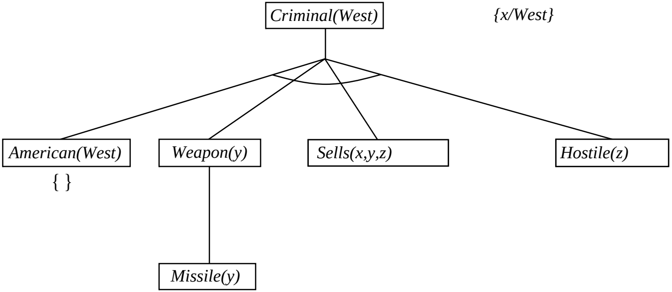
Contoh Backward Chaining FOL



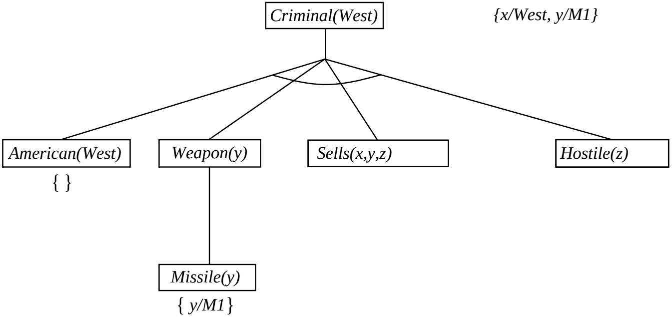
Contoh Backward Chaining FOL



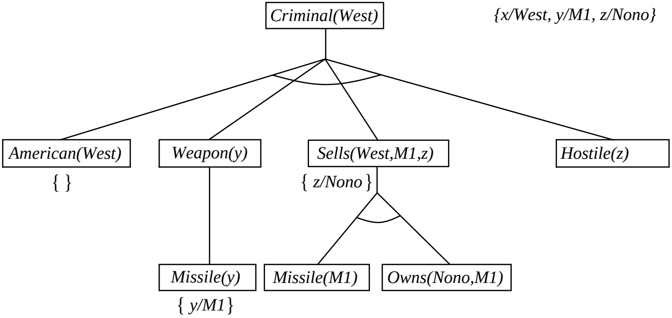
Contoh Backward Chaining FOL



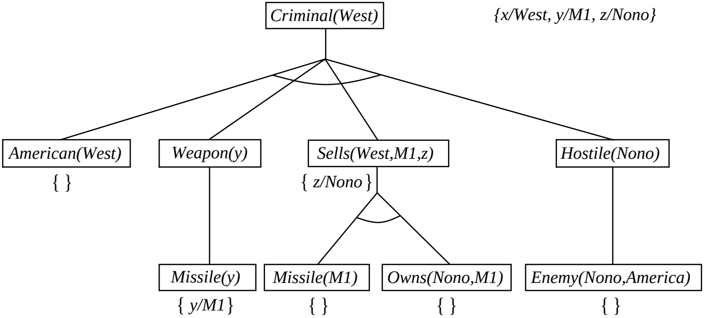
Contoh Backward Chaining FOL



Contoh Backward Chaining FOL



Contoh Backward Chaining FOL



Sifat Backward Chaining

- Depth-first search:
  - linear space complexity ☺
  - incomplete (infinite loop) ☹
  - repeated state ☹
- Prinsip dasar Logic Programming

Outline

Apakah Logic Programming?

Pemrograman dengan logika

Program = deklarasi fakta yang bernilai true
Execution = proses inference (backward chaining)

	Logic programming	Programming biasa
1.	Identifikasi masalah	Identifikasi masalah
2.	Kumpulkan informasi	Kumpulkan informasi
3.	Istirahat	Pikirkan solusinya
4.	Repr. informasi dalam *KB*	Tulis program (source code)
5.	Repr. masalah sebagai *query*	Repr. masalah sebagai data
6.	ASK(*KB*,*query*)	Jalankan program terhadap data
7.	Cari fakta salah	Debugging

(Harusnya) lebih mudah men-debug *Ibukota*(*Bukittinggi*, *Indonesia*) daripada $x = x + 2$!

Prolog

- Dasar: backward chaining pada Horn clause
- Banyak dipakai (EU,JAP "5th gen")
- Program berupa himpunan *clause*:
  `head :- literal`$_1$`, ... literal`$_n$`.`

Contoh

$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$

```
criminal(X):-american(X),weapon(Y),sells(X,Y,Z),hostile(Z).
```

- Aspek "non-logic" dari Prolog:
  - BC berjalan atas-ke-bawah (rule) dan kiri-ke-kanan (clause)
  - Ada predikat untuk aritmetika:: `X is Y*Z+3`
  - Ada clause dengan "efek samping": I/O
  - Closed-world assumption: "negation as failure":
    `hidup(X) :- not mati(X).`
    `hidup(anto)` "benar" jika `mati(anto)` "salah".

Contoh Prolog

Implementasi DFS pada state *X*

```
leluhur(X,Y) :- orangtua(X,Y).
leluhur(X,Z) :- leluhur(X,Y),orangtua(Y,Z).
```

Meng-append dua list

```
append([],Y,Y).
append([X|L],Y,[X|Z]) :- append(L,Y,Z).

query:  append([1,2],[3,4],X) ?
answers:  X=[1,2,3,4]

query:  append(A,B,[1,2]) ?
answers:  A=[] B=[1,2]
A=[1] B=[2]
A=[1,2] B=[]
```

Outline

Resolution pada FOL

Resolution inference rule pada FOL (lifting resolution PL):

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{(\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)\theta}$$

di mana $\text{UNIFY}(\ell_i, \neg m_j) = \theta$.

Contoh:

$$\frac{\begin{array}{l}\neg Kaya(x) \vee Sedih(x) \\ Kaya(Anto)\end{array}}{Sedih(Anto)}$$

di mana $\theta = \{x/Anto\}$

Gunakan resolution rule pada $CNF(KB \wedge \neg\alpha)$: complete untuk FOL

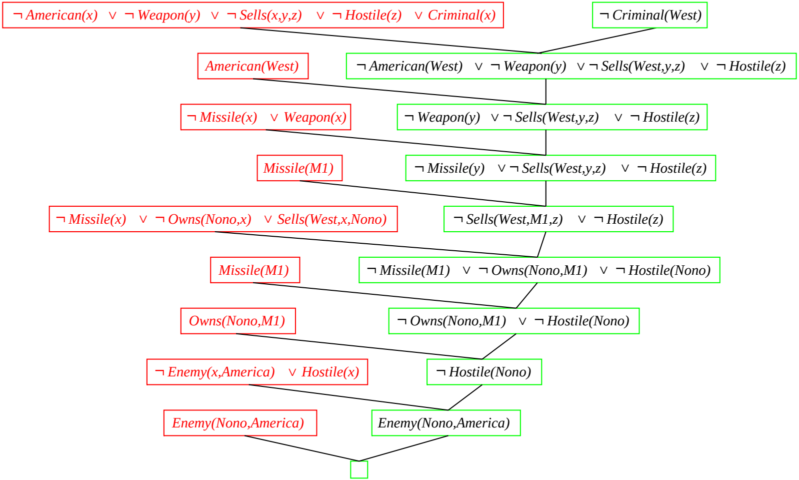
Mengubah FOL ke CNF

"Everyone who loves all animals is loved by someone:"

$\forall x \; [\forall y \; Animal(y) \implies Loves(x,y)] \implies [\exists y \; Loves(y,x)]$

1. Eliminasi implikasi dan biimplikasi
   $\forall x \; [\neg\forall y \; \neg Animal(y) \lor Loves(x,y)] \lor [\exists y \; Loves(y,x)]$
2. Pindahkan $\neg$ ke "dalam": $\neg\forall x, p \equiv \exists x \; \neg p$, $\neg\exists x, p \equiv \forall x \; \neg p$:
   $\forall x \; [\exists y \; \neg(\neg Animal(y) \lor Loves(x,y))] \lor [\exists y \; Loves(y,x)]$
   $\forall x \; [\exists y \; \neg\neg Animal(y) \land \neg Loves(x,y)] \lor [\exists y \; Loves(y,x)]$
   $\forall x \; [\exists y \; Animal(y) \land \neg Loves(x,y)] \lor [\exists y \; Loves(y,x)]$
3. Standardize variables: setiap quantifier variable-nya beda
   $\forall x \; [\exists y \; Animal(y) \land \neg Loves(x,y)] \lor [\exists z \; Loves(z,x)]$
4. Skolemize: generalisasi existential instantiation. $\exists x$ diganti Skolem function universal quantified variable di "luar":
   $\forall x \; [Animal(F(x)) \land \neg Loves(x,F(x))] \lor Loves(G(x),x)$
5. Buang universal quantifiers:
   $[Animal(F(x)) \land \neg Loves(x,F(x))] \lor Loves(G(x),x)$
6. Distribusi $\land$ over $\lor$:
   $[Animal(F(x)) \lor Loves(G(x),x)] \land [\neg Loves(x,F(x)) \lor Loves(G(x),x)]$

Contoh pembuktian dengan resolution



Outline

Ringkasan

- 60’an: Inference FOL melalui PL
- Unification
- Generalized Modus Ponens: FC + BC
- Logic programming: Prolog
- Resolution