LESSON 16 :

INFERENSI PADA FOL

6.2.9 Pembuktian sebagai pencarian

Sampai saat ini kita telah menghasilkan bukti seakan akan kita menemukannya secara ajaib. Kita memberikan formula dan menunjukan pembuktian kepada hasil yang diinginkan. Kita tidak menunjukan bagaimana bukti itu diturunkan.

Kita sekarang akan menunjukan bagaimana sebenarnya menghasilkan bukti itu, dan dalam proses ini kita juga menekankan bertapa terikat dengan dekatnya antar teorema pembuktian dan pencarian.

Prosedur pembuktian umum

Kita menggunakan resolusi binary (kita merepresentasikan klausa sebagai set set) dan merepresentasikan pembuktian sebagai sebuah pohon pencarian. Di dalam tree ini node menyatakan klausa. Kita mulai dengan dua set yang saling bergabung secara disjoin dari klausa INITIAL dan OTHER

1. Kita membuat sebuah node START dan memperkenalkan hyperarc dari START kepada node node baru, yang setiap nodenya melambangkan elemen elemen yang berbeda dari INITIAL. Kita menaruh semua set ini ke dalam sebuah set OPEN. Node node ini kita sebut sebagai AND node;
2. Jika OPEN kosong, Pencarian kita berakhir, jika tidak kita menyingkirkan sebuah elemen N pada OPEN menggunakan sebuah fungsi pemilihan yaitu SELECT.
3. Jika N adalah sebuah node AND, kita gabungkan N dengan sebuah hyperarc terhadap node node N1….Nm, satu untuk semua literal yang terdapat di klausa C direpresentasikan di N. node node ini juga dilabelkan dengan C dan disebut OR node, semua node ini kemudian ditempatkan pada OPEN.

   [Keterangan 1: Pada kasus dimana C terdiri dari satu literal saja, kita bisa katakana bahwa N itu sekarang sebuah OR Node]

   [Keterangan 2 : Kita bisa menentukan untuk tidak menciptakan semua node N1…Nm dalam satu buah aksi namun sebaliknya memilih salah satu literal dari C dan menciptakan node Ni yang merepresentasikan C dengan pilihan literal ini. Ni kemudian dimasukan ke dalam OPEN. N juga kembali ke OPEN jika belum semua literalnya digunakan. Aturan yang digunakan untuk memilih literal di dalam C disebut sebagai Aturan pemilihan]

   Ulangi langkah ke 2

4. Jika N adalah OR node, katakanlah berkorepondensi dengan literal ke I dari klausa C, kita mempertimbangkan semua kemungkinan jalan dengan menerapkan resolusi binary antara C dan klausa dari set OTHER, menyelesaikan literal ke I dari klausa C.

Bila D1… Dp adalah klausa hasil. Kita merepresentasikan setiap klausa Dj dengan AND node N(Dj) seperti pada langkah 1. Kita menaruh arc dari N kepada N(Dj). Kita juga menyiapkan OPEN menjadi Next_OPEN(OPEN, C, {D1…Dp}). Kita menjadikan OTHERS menjadi Next_OTHER(OTHERS,C,{D1…Dp}).Jika di dalam pohon pencarian kita memiliki sebuah hyperpath(sebuah path yang mencakup hyperarc) dimulai dari START yang meninggalkan semua label{} , maka kita berhenti dengan baik

[Keterangan 3 : Selain dari menyelesaikan C dengan literal ke I nya dengan semua kemungkinan elemen dari OTHERS, untuk memilih satu komponen yang compatible dari OTHERS dan untuk menyelesaikan C denganya, menaruh resolven dan C kembali pada OPEN, Kita akan menyebut aturan untuk memilih element dari OTHERS sebagai aturan pencarian]

Dalam procedur pembuktian kita telah membiarkan beberapa hal belum ditentukan :

- Kumpulan set INITIAL dan OTHERS
- Fungsi memilih (SELECT) yang dengannya kita memilih sebuah node untuk diexpand
- Fungsi NEXT_OPEN untuk menghitung nilai selanjutnya dari OPEN
- Fungsi NEXT_OTHERS untuk menghitung nilai selanjutnya dari OTHERS.

Tidak ada jaminan dengan pilihan INITIAL,OTHERS, SELECT, NEXT_OPEN, NEXT_OTHERS apapun bahwa hasil teorema pembuktian akan lengkap (dalam arti bahwa semua yang bisa dibuktikan akan dibuktikan oleh cara pembuktian ini).

Contoh :

Anggaplah kita ingin membuktikan :

1. NOT P(X) OR NOT R(X)
   Tidak konsisten dengan set klausa berikut :
2. NOT S(x) OR H(x)
3. NOT S(X) OR R(X)
4. S(b)
5. P(b)

Dibawah ini adalah pilihan yang mungkin :

INITIAL : {1.} terdiri dari klausa yang mewakili negasi dari goal.

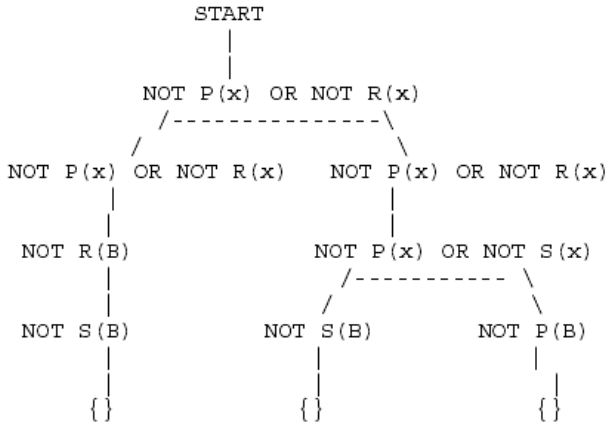
OTHERS : {2.3.4.5} terdiri dari axioma non-logikal

SELECT : kita menggunakan OPEN sebagai FIFO, dengan kata lain melakukan breadth first Search

NEXT_OPEN : membuat NEXT_OPEN(OPEN,C,{D1…Dp}) kepada gabungan dari OPEN, {C}, dan {D1,…Dp}

NEXT_OTHERS : fungsi ini membiarkan OTHERS tak berubah.

Sehingga pohon pencarian : (kita mengarisbawahi AND nodes dan setiap arc keluar diasumsikan membentuk sebuah hyperlink tunggal).



Pada setiap OR node kita menerapkan resolusi antara klausa saat ini dengan literal yang sedang dipilih dan semua component OTHERS yang memungkinkan.

Contoh :

Contoh dibawah ini menggunakan bentuk horn clause dalam logika preposisional. Disini digunakan notasi yang umum digunakan pada prolog, kita merepresentasikan implikasi :

A1 AND A2 AND… AND An IMPLIES A

Sebagai

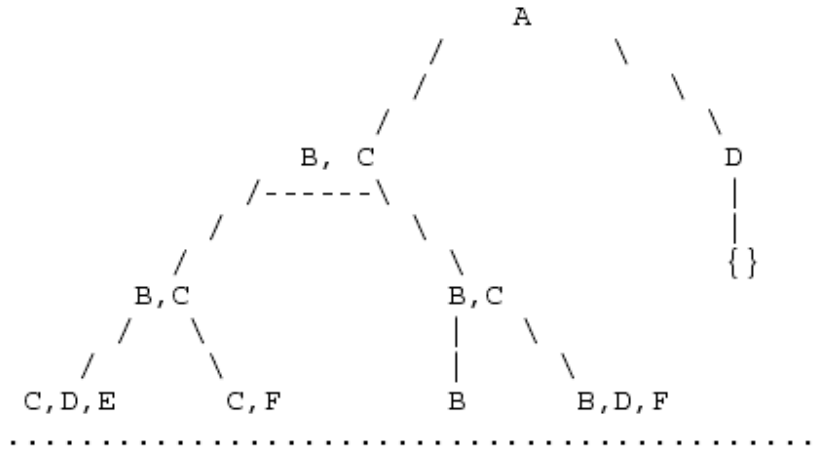
A <= A1,A2,….An

Masalah kita adalah sebagai berikut :

1. A . ini adalah yang ingin kita buktikan
2. A <= B, C
3. A <=D
4. B <= D, E

5. B <= F
6. C <=
7. C <= D, F
8. D <=
9. F <=

Sekarang kita memiliki sebuah pohon pencarian yang secara partial merepresentasikan permasalahan. Kita tidak menerapkan breadth-first search karena kita ingin melihat seberapa besar akibat dari pemilihan fungsi untuk SELECT.



Bila pohon terus di expand secara DFS maka akan menghasilkan pohon pencarian yang sementara BFS akan menurunkan D dari A dan menurunkan {} dari D. dalam kasus yang lain beberapa strategi pencarian lain akan lebih tepat. Seperti yang akan kita lihat dibawah ini

6.2.10 Beberapa strategi pencarian

Dari awal tahun 60an orang orang telah mencari strategi yang akan menyederhanakan masalah pencarian dalam teorema pembuktian. Dibawah ini adalah beberapa strategi pencarian yang disarankan

6.2.10.1 Preferensi unit

Ketika kita menerapkan resolusi jika pada salah satu premis adalah sebuah klausa unit (memiliki literal tunggal). Maka hasil resolusi itu akan memiliki literal yang berkurang satu dari premis terbesar, sehingga dengan demikian semakin mendekati tujuan {}. Maka tampaklah baik untuk menggunakan resolusi pada klausa klausa yang salah satunya adalah sebuah klausa unit. Preferensi unit ini diterapkan ketika memilih dari set OPEN (daun dari pohon pencarian)dan juga saat ketika berada di OR node dan kita memilih elemen dari OTHERS untuk diresolusikan.

6.2.10.2 kumpulan strategi pendukung

Ketika kita menggunakan kumpulan strategi pendukung kita memiliki :

NEXT-OPEN (OPEN, C, {D1,..,..Dn}) adalah sebuah gabungan dari OPEN, {C}, dan {D1….Dn}

NEXT-OTHERS(OTHERS,C , {D1,…Dn})adalah OTHERS

Dengan kata kata yang sederhana, kumpulan strategi pendukung ini menggunakan OPEN sebagai set pendukung (support set). Setiap penggunaan resolusi memiliki premis salah satu elemen dari set pendukung dan menambahkan premis tersebut dan resolventnya ke set pendukung

Biasanya set awal / INITIAL set (set pendukung awal) terdiri dari semua klausa yang didapat dengan menegasikan teorema yang dituju. Kumpulan strategi pendukung dikatakan lengkap apabila :

Isi kumpulan klausa pada OTHERS bersifat satisfiable. Dalam kasus ini kita diberikan sebuah set axiom non logical dan kita gunakan sebagai OTHERS.

6.2.10.3 Resolusi Input

Dalam resolusi input :

INITIAL terdiri dari gabungan antara negasi dari teorema dan kumpulan axiom non logika.

OTHERS biasanya adalah kumpulan logika non-axiom

NEXT-OPEN(OPEN, C, {D1...Dp}) adalah gabungan dari OTHERS , {C}, dan {D1,...Dp}

Dengan kata lain dalam setiap resolusi salah satu premisnya adalah salah satu dari klausa awal

Secara umum resolusi input tidak lengkap, seperti yang dapat dilihat dalam kumpulan klausa yang bersifat unsatisfiable (dari Nilson) dimana kita tidak dapat menurunkan FALSE menggunakan resolusi input :

- Q(u) OR P(A)
- NOT Q(w) OR P(w)
- NOT Q(x) OR NOT P(x)
- Q(y) OR NOT P(y)

[anda bisa menggunakan kumpulan klausa sebagai OPEN dan OTHERS.]

Resolusi input bersifat lengkap pada kasus dimana semua klausa adalah dalam bentuk klausa horn

6.2.10.4 Resolusi linear

Resolusi linear atau yang dikenal juga sebagai *Ancestry-Filtered Form strategy*, adalah generallisasi dari resolusi input. Pengeneralisasian adalah bahwa untuk tiap resolusi salah satu klausa adalah klausa awal atau adalah pendahulu dari klausa yang kedua pada pohon pencarian. Resolusi linear bersifat lengkap.

6.2.10.5 resolusi SLD

Selected linear resolution for Definite Clause (resolusi linear terpilih untuk klausa yang terbatas), atau resolusi SLD, adalah dasar bagi prosedur pembuktian untuk theorema yang berada dalam bentuk konjungsi dari literal positif (konjungsi ini disebut goal) bila diberikan axiom non logikal yang adalah klausa yang terbatas.

Lebih tepatnya :

Goal adalah konjungsi dari literal positif

Aturan pemilihan adalah metoda untuk memillih salah satu literal dari goal (literal perttama, atau literal terakhir atau sebagainya).

Dalam resolusi SLD, bila diberikan suatu goal G = A1...An, klausa terbatas C: A <= B1 .. Bm, dan sebuah sub goal Ai dipilih dari G menggunakan suatu aturan pemilihan S , dimana Ai dan A dapat disatukan dengan MGU's , hasil resolusi dari G dan C dengan S adalah :

(A1...Ai-1 B1...Bm Ai+1...An)S

Sebuah turunan SLD adalah dari sebuah goal G dengan aturan pemilihan S dan sebuah kumpulan klausa terbatas P, adalah urutan triplet (Gi,Ci,Si) dengan i dari 0 sampai k dimana

G0 adalah G

Untuk setiap i > 0, C1 didapat dari sebuah klausa P dengan mengganti semua variablenya dengan variable baru.

Gi+1 adalah resolven dari SLD untuk Gi dan Ci dengan penggunaan S dengan MGU Si

Refutasi dari sebuah goal G bila diberikan klausa P dan aturan pemilihan S, adalah sebuah turunan SLD dari G bila diberikan P dan S dimana goal terakhir adalah sebuah himpunan kosong. Jika s1....sk adalah MGU yang digunakan di dalam refutasi, maka s = si,s2,..sk adalah sebuah subtitusi yang terbatas pada variable G, yang membuat G bernilai benar setiap kali kalusa pada P benar.

Goal G berhasil untuk aturan pemilihan S dan sebuah kumpulan klausa P jika ia memilliki refutasi untuk P dan S maka bila tidak ia dikatakan gagal.

Theorema : Resolusi SLD benar dan lengkap untuk goal yang bersifat konjungtif dan positif dan untuk klausa yang tertentu.

Sebuah konsekuensi yang penting dari teorema ini adalah bahwa ia bernilai benar apapun aturan pemilihan yang kita gunakan untuk memilih literal pada goal. Oleh sebab itu kita bisa memilih literal sekehendak kita, untuk instance dari kiri ke kanan. Aspek lain yang juga penting adalah bahwa subtitusi s = s1,s2,...sn memberikan kita sebuah metode untuk menemukan individu yang memenuhi tujuan dan struktur yang memuaskan klausa pada P.

Tidak ada yang disebutkan dalam SLD mengenai aturan apa yang seharusnya digunakan untuk memilih klausa P untuk menyelesaikan literal saat ini dengan goal saat ini (aturan seperti ini disebut aturan pencarian)

Contoh :

Bila diberikan klausa berikut :

- WOMAN(MOTHER(v)) ; Every mother is a woman
- GOOD(HUSBAND(ANN)) ; The husband of Ann is good
- GOOD(z) <= LIKES(MOTHER(z), z); if z likes his mother then z is good

dan kita ingin mencari wanita yang menyukai suami orang lain.

Maka goal dapat kita sebutkan sebagai :

- WOMAN(x),LIKES(x,HUSBAND(y))

Perhatikanlah bahwa variable x dan variable y adalah dengan kuantifier eksistensial.

Maka turunan SLDnya adalah :

```
((WOMAN(x),LIKES(x,HUSBAND(y))), WOMAN(MOTHER(v)), [MOTHER(v)/x])
((LIKES(MOTHER(v),HUSBAND(y))), GOOD(z) <= LIKES(MOTHER(z), z),
                                [HUSBAND(y)/v, HUSBAND(y)/z])
((GOOD(HUSBAND(y))), GOOD(HUSBAND(ANN)), [ANN/y])
{}
```

dan hasil subtitusinya adalah :

[MOTHER(HUSBAND(ANN))/x, ANN/y]

6.2.11 proses penalaran non monotonic

First order logic dan inferensi yang dilakukan padanya adalah contoh dari proses penalaran monotonic

Dalam penalaran monotonik jika kita mempebesar satu set axiom kita tidak dapat mengambil assertion dan axiom yang ada.

Manusia tidak mengikuti penalaran secara monotonik ini

Kita perlu bisa melompat ke kesimpulan untuk dapat merencanakan, dan lebih dasarnya lagi unutk bertahan hidup

Kita tidak bisa mengantisipasikan semua kemungkinan dari rencana kita

Kita harus dapat mengambil asumsi untuk permasalahan yang tidak kita ketahui.

6.2.11.1 penalaran default.

Hal ini sangatlah umum pada penalaran non monotonik, disini kita ingin mengambil kesimpulan berdasarkan pada apa yang paling mungkin benar.

Kita telah melihat contoh dari hal ini dan beberapa cara untuk merepresentasikan pengetahuan ini,

Kita akan membahas dua pendekatan untuk melakukan ini :

- Non monotonic logic
- Default logic.

Janganlah merasa binggung dengan label non monotonik dan default yang digunakan pada penalaran dan sebuah logika tertentu. Penalaran non monotonic adalah representasi umum dari sebuah kelas penalaran. Logika non monotonik () adalah sebuah teori tersendiri. Hal yang sama berlaku juga untuk penalaran default dan logika default.

Logika non monotonik

Ini tidaklah lain hanya sebuah perkembangan dari first order predicate logic yang menyangkut sebuah operator modal M . tujuan hal ini adalah untuk memungkinkan adanya konsistensi.

Sebagai contoh :

$$\forall x\ play_{instrument(x)} \wedge M_{improvise(x)} \rightarrow jazz\_musician(x)$$

Menyatakan bahwa untuk semua x bila x mampu memainkan alat music dan fakta bahwa x dapat berimprovisasi konsisten dengan semua fakta lain maka dapat disimpulkan x adalah seorang musisi jazz.

Bagaimana kita dapat mendefinisikan konsistensi ?

Salah satu jawaban umum (sesuai dengan notasi Prolog) adalah :

Untuk menunjukan bahwa suatu fakta P itu benar cobalah buktikan ~P, jika kita gagal maka dapat dikatakan p itu konsisten (karena ~P adalah false).

Namun cobalah pertimbangkan contoh assertion yang terkenal ini mengenai President Nixon :

$$\forall x\ Republician(x) \wedge M_{\sim Pasifist(x)} \rightarrow \sim Pasifist(x)$$

$$\forall x\ Quaker(x) \wedge M_{\sim Pasifist(x)} \rightarrow Pasifist(x)$$

Dua pernyataan ini menyatakan bahwa Quaker cenderung bersifat pasif dan republician tidak.

Tapi Nixon termasuk Quaker dan Republician , jadi kita kita masukan fakta :

Quaker (Nixon)

Republician(Nixon)

Ini akan berujung pada knowledge base kita menjadi tidak konsisten

Logika Default

Logika default memperkenalkan suatu aturan inferensi baru :

$$\frac{A : B}{C}$$

Yang menyatakan bahwa A dapat diturunkan, dan adalah konsisten untuk mengasumsikan B dan lalu menyimpulkan C

Ini serupa dengan logika non monotonik, tapi dengan beberapa perbedaan :

- Aturan inferensi baru digunakan untuk menghitung kumpulan tambahan yang mungkin. Jadi pada kasus Nixon logika default dapat mendukung kedua input karena ia tidak mengatakan bagaimana cara memilih diantaranya – hal ini akan bergantung pada cara menginferensikannya.
- Dalam logika default setiap ekspresi monotonic adalah aturan inferensi dan bukan ekspresi

6.2.11.2 Circumscription

Circumscription adalah sebuah aturan conjecture yang memungkinkan anda meloncat pada kesimpulan bahwa sebuah obyek yang memiliki properti tertentu (P) adalah semua obyek yang memiliki properti tersebut.

Circumscription juga dapat menghadapai penalaran default

Anggaplah kita memiliki : bird(Tweety)

$$\forall \mathrm{x}\ \mathrm{Pengguin(x)} \rightarrow \mathrm{bird(x)}$$

$$\forall \mathrm{x}\ \mathrm{Pengguin(x)} \rightarrow \sim \mathrm{Flies(x)}$$

Maka kita ingin menambahkan pada umumnya burung dapat terbang.

Pada circumscription frase tersebut akan dinyatakan sebagai berikut :

A bird will fly if it is not abnormal

Dan karenanya dapat dinyatakan sebagai :

$$\forall x\, \mathrm{bird}(x) \wedge \sim \mathrm{abnormal}(x) \rightarrow \mathrm{Flies}(x)$$

Namun, hal ini tidak cukup

Kita tak bisa menyimpulkan

Flies (Tweety)

Karena kita tidak bisa membuktikan

~abnormal(Tweety)

Disinilah saatnya kita menerapkan circumscription., dan dalam kasus ini :

Kita akan mengasumsikan bahwa semua hal yang ditunjukan sebagai abnormal adalah semua hal yang abnormal.

Maka kita bisa menuliskan kembali aturan default kita menjadi :

$$\forall x\, \mathrm{bird}(x) \wedge \sim \mathrm{abnormal}(x) \rightarrow \mathrm{Flies}(x)$$

Dan menambahkan fakta berikut :

$$\forall x \sim \mathrm{abnormal}(x)$$

Karena tidak ada yang ditunjukan sebagai abnormal sekarang kita dapat tambahkan fakta :

Pengguin(Tweety)

Dan dapat membuktikan

Abnormal(Tweety)

Jika kita ingin membuktikan ke abnormalan kita akan tambahkan kalimat berikut :

Pengguin(Tweety) adalah suatu yang abnormal

$$\forall x\, \mathrm{abnormal}(x) \rightarrow \mathrm{Pengguin}(x)$$

Perhatikanlah perbedaan antara circumscription dan logika defaut

Default adalah kalimat di dalam bahasa itu sendiri dan bukan tambahan aturan inferensi

6.2.11.3 Sistem pemeliharaan kebenaran

Ada banyak cara untuk sistem pemeliharaan kebenaran yang telah dibuat sebagai alat untuk menerapkan sistem penalaran non monotonik

Dasarnya sistem ini :

Semuanya melakukan sejenis backtraking yang terarah pada ketergantungan

Penambahan fakta terhubung dengan jaringan ketergantungan

Sistem pemeliharaan kebenaran berdasarkan pembenaran (Justification-based Truth maintenance system / JTMS)

- Hal ini adalah sistem pemeliharaan kebenaran sederhana karena tidak mengetahui tentang struktur dari input itu sendiri
- Setiap fakta yang didukung mempunyai pembenarannya
- Setiap pembenaran memiliki dua bagian :
    - Sebuah IN-list – yang mendukung fakta yang dipercaya saat ini
    - Sebuah OUT-list – yang mendukung fakta yang belum dipercaya saat ini
- Sebuah pernyataan dihubungkan dengan pembenaran dengan sebuah panah
- Satu pernyataan dapat terlingkup dalam pembenaran yang lain sehingga menciptakan suatu jaringan
- Suatu pernyataan dapat dilabelkan dengan sebuah tingkat kepercayaan
- Sebuah pernyataan dikatakan valid apabila semua pernyataan di IN-list dipercaya dan tidak ada satupun pernyataan di OUT-list dipercaya
- Sebuah pernyataan dikatakan non-monotonik jika OUT-listnya tidak kosong atau semua pernyataan pada IN-list bersifat non-monotonik.

Sistem pemeliharaan kebenaran berdasarkan logika (LTMS)

Serupa dengan sistem pembenaran berdasarkan pembenaran kecuali :

- Node mengasumsikan tidak adanya hubungan antara mereka kecuali mereka yang secara explicit di nyatakan pada pembenarannya.
- Sistem pemeliharan kebenaran dengan pembenaran dapat merepresentasikan P dan ~P secara bersamaan. Sebuah LTMS akan menunjukan adanya kontradiksi disini
- Jika hal ini terjadi maka jaringan harus dibuat kembali

Sistem pemeliharaan kebenaran berdasarkan asumsi (ATMS)

- JTMS dan LTMS mengejar satu baris pertimbangan untuk tiap waktu dan backtrack apabila diperlukan. – depth first search
- ATMS mempertahankan jalur alternative secara parallel – breadth first search
- Backtraking dihindari dengan mempertahan banyak konteks
- Bagaimanapun juga saat penalaran terjadi kontradiksi akan muncul dan ATMS dapat dipotong
    - Carilah pernyataan dengan pembenaran yang tidak valid

Pertanyaan :

1.a mempertimbangkan bahwa sebuah sistem first order logic yang menggunakan dua predikat bernama BIG dan SMALL, kumpulan object yang diberikan adalah {A,B}. enumerasikanlah semua model yang mungkin terjadi untuk kasus ini.

1.b untuk setiap kalimat berikut, identifikasikan model dimana kalimatnya benar

a. $Big(A) \wedge Big(B)$
b. $Big(A) \vee Big(B)$
c. $\forall x\ Big(x)$
d. $\forall x \neg Big(x)$
e. $\exists x\ Big(x)$
f. $\exists x \neg Big(x)$
g. $\forall x\ Big(x) \wedge Small(x)$
h. $\forall x\ Big(x) \vee Small(x)$
i. $\forall x\ Big(x) \Rightarrow \neg Small(x)$

1.c Tentukan apakah ekspresi p dan q saling bersatu dengan lainnya dalam tiap kasus ini. Jika iya, berikan unifier yang paling umum, jika tidak berikanlah penjelasan singkat (asumsikan bahwa kata berhuruf kapital adalah object, predikat, fungsi, dan konstanta. Sedangkan berhuruf kecil adalah variable)

a. $p = F(G(v), H(u, v)); q = F(w, J(x, y))$
b. $p = F(x, F(u, x)); q = F(F(y, A), F(z, F(B, z)))$
c. $p = F(x_1, G(x_2, x_3), x_2, B); q = F(G(H(A, x_5), x_2), x_1, H(A, x_4), x_4)$

1.d taruhlah FOL berikut ke dalam bentuk CNF :

a. $\forall x\ \forall y\ ((P(x) \wedge Q(y)) \Rightarrow \exists z\ R(x, y, z))$
b. $\exists x\ \forall y\ \exists z\ (P(x) \Rightarrow (Q(y) \Rightarrow R(z)))$

2. Ubahlah kalimat berbahasa inggris ini menjadi kalimat dalam bentuk FOL

a. every boy or girl is a child
b. every child gets a doll or a train or a lump of coal
c. no boy gets any doll
d. no child who is good gets any lump of coal

e. Jack is a boy

3. Menggunakan kelima axioma diatas, bentuklah pembuktian dengan refutasi dan resolusi pada pernyataan :

   "if Jack doesn't get a train, then Jack is not a good boy"

4. Pertimbangkanlah kumpulan axioma di bawah ini :

i. Everyone who loves someone who loves them back is happy

ii. Mary loves everyone.

iii. There is someone who loves Mary.

Buktikan lah dari kalimat kalimat diatas : Marry is happy

Solusi :

1. a

$M_0 = \{\}$
$M_1 = \{\text{Big(A)}\}$
$M_2 = \{\text{Big(B)}\},$
$M_3 = \{\text{Small(A)}\}$
$M_4 = \{\text{Small(B)}\}$
$M_5 = \{\text{Big(A), Big(B)}\}$
$M_6 = \{\text{Big(A),Small(A)}\}$
$M_7 = \{\text{Big(A), Small(B)}\}$
$M_8 = \{\text{Small(A), Big(B)}\}$
$M_9 = \{\text{Big(B),Small(B)}\}$
$M_{10} = \{\text{Small(A), Small(B)}\}$
$M_{11} = \{\text{Big(A), Big(B), Small(A)}\}$
$M_{12} = \{\text{Big(A), Big(B), Small(B)}\}$
$M_{13} = \{\text{Small(A), Small(B), Big(A)}\}$
$M_{14} = \{\text{Small(A), Small(B), Big(B)}\}$
$M_{15} = \{\text{Big(A), Big(B), Small(A), Small(B)}\}$

1.b

(a) $Big(A) \wedge Big(B)$: $M_5 M_{11} M_{12} M_{15}$

(b) $Big(A) \vee Big(B)$: $M_1 M_2 M_5 M_6 M_7 M_8 M_9 M_{11} M_{12} M_{13} M_{14} M_{15}$

(c) $\forall x Big(x)$(equivalent to $Big(A) \wedge Big(B)$) : $M_5 M_{11} M_{12} M_{15}$

(d) $\forall x \neg Big(x)$(equivalent to $\neg Big(A) \wedge \neg Big(B)$) : $M_0 M_3 M_4 M_{10}$

(e) $\exists x Big(x)$(equivalent to $Big(A) \vee Big(B)$) : $M_1 M_2 M_5 M_6 M_7 M_8 M_9 M_{11} M_{12} M_{13} M_{14} M_{15}$

(f) $\exists x \neg Big(x)$(equivalent to $\neg Big(A) \vee \neg Big(B)$ which is equivalent to $\neg(Big(A) \wedge Big(B))$) :
$M_0 M_1 M_2 M_3 M_4 M_6 M_7 M_8 M_9 M_{10} M_{13} M_{14}$

(g) $\forall x Big(x) \wedge Small(x)$(equivalent to $(Big(A) \wedge Small(A)) \wedge (Big(B) \wedge Small(B))$) : M_{15}

(h) $\forall x Big(x) \vee Small(x)$(equivalent to $(Big(A) \vee Small(A)) \wedge (Big(B) \vee Small(B))$) : $M_5 M_7 M_8 M_{10} M_{11} M_{12} M_{13} M_{14} M_{15}$

(i) $\forall x Big(x) \Rightarrow Small(x)$(equivalent to $(\neg Big(A) \vee \neg Small(A)) \wedge (\neg Big(B) \vee \neg Small(B))$) :
$M_0 M_1 M_2 M_3 M_4 M_5 M_7 M_8 M_{10}$

1.c

(a) $p = F(G(v), H(u, v)); q = F(w, J(x, y))$.

Substitute $w/G(v)$: $p = F(G(v), H(u, v)); q = F(G(v), J(x, y))$ but cannot find a unifier for $H(u, v)$ and $J(x, y)$ because they are different objects (predicates, functions, constants).

(b) $p = F(x, F(u, x)); q = F(F(y, A), F(z, F(B, z)))$.

Substitute $x/F(y, A)$: $p = F(F(y, A), F(u, F(y, A))); q = F(F(y, A), F(z, F(B, z)))$.

Substitute u/z: $p = F(F(B, A), F(z, F(y, A))); q = F(F(y, A), F(z, F(B, z)))$.

Substitute y/B and z/A: $p = F(F(B, A), F(A, F(B, A))); q = F(F(B, A), F(A, F(B, A)))$.

So Unify(p, q)=$\{x/F(y, A), u/z, y/B, z/A\}$.

(c) $p = F(x_1, G(x_2, x_3), x_2, B); q = F(G(H(A, x_5), x_2), x_1, H(A, x_4), x_4)$.

Substitute x_4/B: $p = F(x_1, G(x_2, x_3), x_2, B)$; $q = F(G(H(A, x_5), x_2), x_1, H(A, B), B)$.

Substitute $x_1/G(x_2, x_3)$: $p = F(G(x_2, x_3), G(x_2, x_3), x_2, B)$; $q = F(G(H(A, x_5), x_2), G(x_2, x_3), H(A, B), B)$.

Substitute $x_2/H(A, x_5)$: $p = F(G(H(A, x_5), x_3), G(H(A, x_5), x_3), H(A, x_5), B)$;

$q = F(G(H(A, x_5), H(A, x_5)), G(H(A, x_5), x_3), H(A, B), B)$.

Substitute $x_3/H(A, x_5)$: $p = F(G(H(A, x_5), H(A, x_5)), G(H(A, x_5), H(A, x_5)), H(A, x_5), B)$;

$q = F(G(H(A, x_5), H(A, x_5)), G(H(A, x_5), H(A, x_5)), H(A, B), B)$.

Substitute x_5/B so Unify(p, q)=$\{x_4/B, x_1/G(x_2, x_3), x_2/H(A, x_5), x_3/H(A, x_5), x_5/B\}$.

1.d

(a) $\forall x \forall y((P(x) \wedge Q(y) \Rightarrow \exists z R(x,y,z)) =$
$\forall x \forall y(\neg(P(x) \wedge Q(y)) \vee \exists z R(x,y,z)) =$
$\forall x \forall y(\neg P(x) \vee \neg Q(y) \vee R(x,y,G(x,y))) =$
$\forall x \forall y((P(x) \wedge Q(y) \Rightarrow R(x,y,G(x,y)))$

(b) $\exists x \forall y \exists z(P(x) \Rightarrow$
$(Q(y) \Rightarrow R(z))) =$
$\exists x \forall y \exists z(\neg P(x) \vee (Q(y) \Rightarrow R(z))) =$
$\exists x \forall y \exists z(\neg P(x) \vee (\neg Q(y) \vee R(z))) =$
$\exists x \forall y \exists z(\neg P(x) \vee \neg Q(y) \vee R(z))$ (where B is any constant not used elsewhere) $=$
$\forall y(\neg P(B) \vee \neg Q(y) \vee R(G(y))) =$
$\forall y(P(B) \wedge Q(y)) \Rightarrow R(G(y))$

2.

a. forall x ((boy(x) or girl(x)) -> child(x))
b. forall y (child(y) -> (gets(y,doll) or gets(y,train) or gets(y,coal)))
c. forall w (boy(w) -> !gets(w,doll))
d. forall z ((child(z) and good(z)) -> !gets(z,coal))
e. boy(Jack)

3. pernyataan yang ingin dibuktikan dapat ditulis dalam bentuk FOL sebagai berikut :

! gets (Jack, Train) -> !good(jack)

Pembuktian akan melewati 3 tahapan :

- Negasikan kesimpulan
- Ubah semua pernyataan menjadi bentuk clausal
- Terapkan resolusi kepada klausa sampai mencapai klausa kosong.

Ubah semua pernyataan menjadi bentuk clausal

1. forall x ((boy(x) or girl(x)) -> child(x))
   !(boy(x) or girl(x)) or child(x)
   (!boy(x) and !girl(x)) or child(x)
   (!boy(x) or child(x)) and (!girl(x) or child(x))
2. forall y (child(y) -> (gets(y,doll) or gets(y,train) or gets(y,coal)))
   !child(y) or gets(y,doll) or gets(y,train) or gets(y,coal)
3. forall w (boy(w) -> !gets(w,doll))
   !boy(w) or !gets(w,doll)
4. forall z ((child(z) and good(z)) -> !gets(z,coal))
   !(child(z) and good(z)) or !gets(z,coal)
   !child(z) or !good(z) or !gets(z,coal)
5. boy(Jack)
6. !(!gets(Jack,train) -> !good(Jack)) negated conclusion
   !(gets(Jack,train) or !good(Jack))
   !gets(Jack,train) and good(Jack)

Dalam bentuk CNF

1. (a) !boy(x) or child(x)
   (b) !girl(x) or child(x)
2. !child(y) or gets(y,doll) or gets(y,train) or gets(y,coal)
3. !boy(w) or !gets(w,doll)
4. !child(z) or !good(z) or !gets(z,coal)
5. boy(Jack)
6. (a) !gets(Jack,train)
   (b) good(Jack)

Resolusi :

- 4. !child(z) or !good(z) or !gets(z,coal)
  6.(b). good(Jack)
  ---
  7. !child(Jack) or !gets(Jack,coal) (substituting z by Jack)
- 1.(a). !boy(x) or child(x)
  5. boy(Jack)
  ---
  8. child(Jack) (substituting x by Jack)
- 7. !child(Jack) or !gets(Jack,coal)
  8. child(Jack)
  ---
  9. !gets(Jack,coal)
- 2. !child(y) or gets(y,doll) or gets(y,train) or gets(y,coal)
  8. child(Jack)
  ---
  10. gets(Jack,doll) or gets(Jack,train) or gets(Jack,coal) (substituting y by Jack)
- 9. !gets(Jack,coal)
  10. gets(Jack,doll) or gets(Jack,train) or gets(Jack,coal)
  ---
  11. gets(Jack,doll) or gets(Jack,train)
- 3. !boy(w) or !gets(w,doll)
  5. boy(Jack)
  ---
  12. !gets(Jack,doll) (substituting w by Jack)
- 11. gets(Jack,doll) or gets(Jack,train)
  12. !gets(Jack,doll)
  ---
  13. gets(Jack,train)
- 6.(a). !gets(Jack,train)
  13. gets(Jack,train)
  ---
  14. empty clause

4.

A1: for-all x, for-all y, [loves(x,y) & loves(y,x) = > happy(x)]

A2: for-all z, [loves(mary,z)]

A3: there-is w, [loves(w,mary)]

Maka kesimpulan dapat ditulis sebagai berikut :

C: happy(mary)

Maka pembuktian akan berlangsung sebagai berikut :

Translasikan axioma dan negasi dari kesimpulan dalam bentuk klausal. Tunjukan setiap langkah dari translasi (ingatlah urutan yang diterapkan)

1. **A1:** !loves(x,y) || !loves(y,x) || happy(x)
   note that p = > q is equivalent to !p || q and that !(r & s) is equivalent to !r || !s
2. **A2:** loves(mary,z)
3. **A3:** loves(**a**,mary), where **a** is a new constant symbol
4. **!C:** ! happy(mary)

Lalu terapkan resolusi sampai klausa kosong ditemukan

```
!C: ~~! happy(mary)~~
A1:  !loves(x,y) || !loves(y,x) || ~~happy(x)~~
_____________________________________________  with x=mary
A4:  ~~!loves(mary,y)~~ || !loves(y,mary)
A2: ~~loves(mary,z)~~
_____________________________________________  with z=y
A5:  ~~!loves(y,mary)~~
A3: ~~loves(a,mary)~~
_____________________________________________  with y=a
   empty clause
```

Karena kita mengasumsikan ~happy(mary) menjadikan kesimpulandi A1,A2,A3 maka pernyataan happy(marry) adalah kseimpulan logis dari A1,A2,A3