Logic merupakan jantung dari program, para pemrogram mempunyai keyakinan jika sebuah komputer dapat dibuat mengerti logika, maka komputer dapat dibuat untuk berpikir, karena logika kelihatannya menjadi inti dari kecerdasan.

2.1. Sejarah Singkat Logika

Ahli logika pertama yang dikenal : Aristotle(384-322 BC), filosofi dan saintis alami Yunani. Aristotle telah mengembangkan banyak teori yang dikenal dengan *syllogistic* atau *classical logic.*

Syllogistic logic pada dasarnya bertransaksi dengan penurunan kebenaran (atau yang bersifat salah) dari argumen seorang filosofi. [9)]

Contoh: John is a man
All men used to the boys
Therefore, John used to be a boy

Logikanya adalah: John adalah seorang anak laki-laki sebelum dia menjadi orang dewasa.

Contoh diatas dikonversikan ke *Syllogistic logic*:

J ---> M
all M ---> B
hence: J ---> B

Symbolic logic dimulai dengan G.W. Leibniz(1646-1717), tetapi dilupakan setelah ia meninggal, kemudian seluruh lapangan tersebut dicakup kembali oleh: George Boole(1815-1864) dan logikanya dikenal dengan *Boolean Logic*. Symbolic logic berinteraksi dengan konsep abstraksi ke dalam simbol-simbol dan interkoneksi simbol-simbol oleh operator tertentu. [10)]

Contoh: if P is true
Q is false
Then P or Q is true
P and Q is false

[9)] Herbert Schildt, "Artificial Intelligence Using C", Mc.Graw-Hill, Singapura, 1987, halaman 268.
[10)] IBID [9)], halaman 269.

Dalam simbolic logic, terdapat dua perbedaan :

1. *Propotional Logic* : bertransaksi dengan kebenaran atau kesalahan dari sebuah proposition.

2. *Predicate Calculus*: memasukkan hubungan antara obyek-obyek dan kelas-kelas dari obyek.

Karena itu, sistem formal yang memanipulasi kalimat-kalimat standar menurut ketentuan(*rule*) yang dispesifikasikan dengan baik dan mengijinkan beberapa jenis dari *inference*(kesimpulan) yang dibuat. Sebuah sistem merupakan kombinasi dari *proposional logic* atau *proposional calculus* dan *first order predicate calculus*. Kesimpulan yang mendetail dari calculus ini perhatiannya pada mekanisme karakter pengikut. Ada juga suatu klasifikasi dari batas antara kalkulus dan tafsirannya(interpretasinya).

2.2. P*ropositional Logic*

Proposional logic berupa kalimat-kalimat lengkap dari fakta atau kenyataan (*facts/propositions*), seperti :[11)]

"john loves mary" atau "birds fly south in autumn"

Atau bisa dikatakan sebuah *propositional logic* bisa merupakan sebuah proposisi adalah kalimat yang terbentuk dengan sendirinya apakah bernilai *true*(benar) atau *false*(salah).

Propositional Logic menggunakan operator-operator untuk menghubungkan proposisi-proposisi(*propositions*) dalam bentuk ungkapan/ ekspresi / *expression* berupa kata penyambung *logika (Logical connectives*), yaitu berdasarkan tingkatan/*precedences*, sebagai berikut:

tingkatan	Tanda/Sign	Arti [12)]
1.	¬	NOT(*Negation*)
2.	∧	AND(*Conjuction*)
3.	∨	OR(*Disjunction*)
4.	→	IF ... THEN ...(Implikasi)
5.	↔	IF and only IF...then... (iff)(*Double Implication*)
6.	≡	Assignment (*Equivalent*)

[11)] TR. Addis, "Designing Knowledge-Based Systems", KOGAN PAGE, 1986, halaman 182.
[12)] IBID [11)], halaman 183

Contoh : 2 proposisi : P dan Q yang direpresentasikan sebagai ekspresi logika dengan menggunakan *logical connectives* dalam suatu tabel kebenaran, berikut ini:

P	Q	$P \wedge Q$	$\neg P$	$\neg P \vee Q$	$P \rightarrow Q$	$P \leftrightarrow Q$	$(\neg P \vee Q) \equiv (P \rightarrow Q)$
T	T	T	F	T	T	T	T
T	F	F	F	F	F	F	T
F	T	F	T	T	T	F	T
F	F	F	T	T	T	T	T

Ekpresi-ekspresi dibentuk menurut sebuah tata bahasa/grammar sederhana, dan ekspresi yang sesuai dengan tata bahasa ini disebut *well formed formulae(wffs).* Tanda kurung*(parantheses)* digunakan untuk membuat jelas urutan dari penempatan nilai kebenaran, jika tidak yang lain jelas. Suatu *Well Formed Formulae* merupakan salah satu suatu proposisi atau akan mempunyai salah satu bentuk seperti yang terlihat pada tabel berikut ini:

Jika P adalah sebuah Wff maka not P ($\neg P$) juga suatu Wff.

Jika P dan Q adalah dua Wffs, maka:

Wff
$(\neg P)$
$(P \wedge Q)$
$(P \vee Q)$
$(P \rightarrow Q)$
$(P \leftrightarrow Q)$

Sebuah Struktur (*atomic*) formula merupakan sebuah operasi *predicate letter* pada sebuah terminal(term) adalah sebuah Wff. Sebuah term merupakan sebuah variabel atau sebuah operasi fungsi pada sabuah variabel pada sebuah variabel.

contoh: f(x,y)

plus (S, 10)

<u>DOMAIN DARI INTERPRETASI (*INTERPRATATION*)</u>

Predicates……..Domain of Relations

Function …….Domain of Mapping

Variabel……..Domain of Constans, Numbers, dan sebagainya.

Kata penyambung cenderung mempunyai arti yang sama dengan bahasa alami mereka, jadi:

jika: [13)]

'*john love Mary's is True'* (P)

maka :

'*john does not love Mary's is False'* ($\neg$P)

dan jika :

'*Birds fly South in Autumn' is True*' (Q)

maka:

'*john loves Mary and Birds fly South in Autumn'* (P$\wedge$Q)

adalah sebuah proposisi gabungan yang bernilai *true.*

Bila mana, keadaan mesti diambil dalam menterjemahkan *Implication*, sejak ia boleh menggunakan suatu kondisi hubungan yang mempunyai sebuah sebab.

Jadi : '***if it is raining then the road are wet'***

Implikasi menghindari bahwa tafsiran memerlukan keduanya didefinisikan secara formal dan mengikutkan transaksi suatu perubahan lingkungan.

Disini bisa saja : John may not always love Mary dan

The road are not always wet.

Sebuah tafsiran (*Interpretation*) dari sebuah Wff merupakan suatu penempatan nilai kebenaran pada proposisi atom komponennya.

No. Tafsiran	It is raining (P)	The roads are wet (Q)	*Wff* P $\rightarrow$ Q
1	*T*	*T*	*T*
2	T	F	F
3	F	*T*	T
4	F	F	T

---

[13] IBID [11] , halaman 184 -185

2.3. *Predicate Calculus*

Kadang disebut *Predicate Logic* adalah penyederhanaan ekstensi *propositional logic*.

PREDICATE CALCULUS → keunggulan : Dalam pendefinisian Semantic(ARTI KATA).

→ Pembuktian kebenaran peraturan–peraturan kesimpulan dengan baik. (*INFERENCE RULE*).

PREDICATE CALCULUS merupakan salah satu dari skema-skema yang digunakan dalam Representasi Pengetahuan.

Anda dapat menganggap ini sebagai sebuah Bahasa Representasi untuk AI.

Setiap Bahasa mempunyai :

(a) SYNTAX

(b) SEMANTIC

Syntax dari *Predicate Calculus* terdiri dari :

Predicate Letters : P, Q, R, P_1, P_2 . . . . .

Function Letters : f, g, h, f_1, f_2 , ...

Variabel *(Variables)* : X, Y, Z, X_1, X_2, ...

Konstanta (*Contants*) : a,b,c,a_1,a_2, . . … .

Semantic adalah arti yang berhubungan dengannya. Setiap dari mereka boleh mempunyai sebuah daerah/domain. Definisi dari sebuah domain disebut sebuah INTERPRETASI dihimpun dari arti kata/*Semantic*.

Dasarnya: *PREDICATE*, dimana beberapa *function* yang mengembalikan sebuah nilai benar /salah tergantung pada argumennya.

Perbedaan dasar antara *Predicate Logic & Propositional Logic* adalah :

Pemisahan attribute dari obyek yang kemungkinan milik atribute, yaitu dalam *predicate calculus* dimungkinkan untuk membentuk sebuah fungsi yang menentukan kesulitan sebuah obyek yang diberikan. Dalam *Propositional Logic* , kita harus membentuk statement baru untuk setiap kasus. Jika yang ada hanya *Predicate Letters* (Dan tanpa variabel *Quantifier,* dan sebagainya) disebut *O_order Predicate calculus* atau *Prepositional calculus*, atau *Sentential calculus.*

Walaupun bentuk *Propositional Logic* dasar untuk kecerdasan dan bahasa komputer, tetapi kita tidak dapat menggunakan bentuk ini dengan sendirinya untuk menyatakan pengetahuan manusia di dunia, karena bentuk ini kurang mampu untuk menunjukkan hubungan antar obyek, bentuk ini terbatas hanya untuk penentuan kebenaran atau kesalahan dari sebuah contoh yang diberikan dan tidak dapat digunakan pada klasifikasi. Satu hal yang penting bahwa *Predicates* dapat memiliki beberapa argument.

Untuk *Propositional Expression* P,Q, & R : [14)]

P_1 . . . . . .~ (~P) ≡ P (*double negative/inverse*)

P_2 . . . . .(P ∨ Q) ≡ (~ P → Q) juga (P → Q) = (~ P∨ Q)

P_3 . . . .Hukum De Morgan 1 : ~ (P ∨ Q) ≡ (-P ∧ ~ Q)

P_4 . . . .Hukum De Morgan 2 : ~ (P ∧ Q) ≡ (-P ∨ ~ Q.)

P_5 . . . .Hukum Distributive 1 : P ∨ (Q ∧ R) ≡ (P ∨ Q) ∧ (P ∨ R)

P_6 . . . .Hukum Distributive 2 : P ∧ (Q ∨ R) ≡ (P ∧ Q) ∨ (P ∧ R)

P_7 . . . Hukum Komutatif 1 : (P ∧ Q) ≡ (Q ∧ P)

P_8 . . . Hukum Komutatif 2 : (P ∨ Q) ≡ (Q ∨ P)

P_9 . . . Hukum Assosiatif 1 : ((P ∧ Q) ∧ R) ≡ (P ∧ (Q ∧ R))

P_{10} . . . Hukum Assosiatif 2 : ((P ∨ Q) ∨ R) ≡ (P ∨ (Q ∨ R))

P_{11} . . . Hukum Kontrapositif : (P → Q) ≡ (~Q → ~ P)

Sebuah jalan yang tepat untuk menggambarkan persamaan logika dari jenis ini adalah sekumpulan *production rules* (yang mengkombinasikan dalam sebuah rule : *forward* dan *backward*) untuk memecahkan persoalan dalam *propositional calculus*. Untuk membuktikan sebuah exspresi Q (*goal*) dari pemberian sebuah Wff tunggal. (data base awal), sebuah proses pemilikan suatu urutan dari akhir persamaan dalam Q akan membuat Q dari Wff asli.

[14)] IBID [11)] , halaman 187

Persamaan adalah langkah-langkah dalam sebuah argumen gabungan dan setiap langkah adalah *valid* (benar) (selalu benar, tidak masalah apa tafsirannya).

Sebuah *Argument* adalah sekumpulan dari Wffs diikuti oleh sebuah Wff tunggal disebut kesimpulan (*Conclusion*).

Argument dapat dirangkai adalah sebuah *argument* tunggal atau *proof*. Sebuah argument lengkap dapat dibuat dalam sebuah kalimat:

‘ **if <** *premise**s*** **>** **then** <*conclusion*> ‘.

Dasar pikiran/alasan kesimpulan

Kesimpulan dikatakan mengikuti secara logika dari dasar pikiran dan tergantung dari proposisi (sebuah argument tetap valid tidak masalah apakah penambahan alasan di suplay) merupakan karakter *monotonic* dari Propositional dan *Predicate Calculus*.

Features monotonic tidak selalu di tampilkan dalam argumentasi alamiah sejak informasi baru (tafsiran) dapat mengubah sebuah kesimpulan. Mekanisme dari logika bukan *monotonic* tidak sepenuhnya dimengerti.

Untuk logika *monotonic* bentuk argumennya disebut

modus ponens: $(P, P \rightarrow Q) \therefore Q$

Untuk logika bukan *monotonic* bentuk argumennya disebut

Modus tollen : $(\sim Q, P \rightarrow Q) \therefore \sim P$

Dan kedua mekanisme tersebut yang Wff dapat di *generated*. Keduanya dapat diperiksa dengan tabel kebenaran untuk mengkonfirmasikan atau menyatakan kalimat-kalimat:

$(P \wedge (P \rightarrow Q) \rightarrow Q$ is valid

dan:

$(\sim Q \wedge (P \rightarrow Q)) \rightarrow \sim P$ is valid

Secara respektif untuk *modus ponens* :

P	Q	P → Q	P ∧ (P → Q)	(P∧(P→))→Q
T	T	T	T	T
T	F	F	F	T
F	T	T	F	T
F	F	T	F	T

Contoh yang lain : (Q, P → Q) ∴ P adalah *a fallacy (not valid)*

Contoh :

- Jika P merupakan *propositional* " hari ini hujan" dan Q merupakan *propositional* " saya ada kerjaan" maka himpunan dari *propositional* (P,Q) mempunyai 4 fungsional berbeda dalam nilai kebenaran (T,F).
- P merupakan kalimat " hujan pada hari selasa", kita dapat membuat predicat cuaca yang menjelaskan sebuah hubungan diantara hari dan cuaca, sebagai berikut:

  Cuaca (selasa , hujan)
- Mengarah pada ketentuan–ketentuan kesimpulan (*Inference Rules*), kita dapat merubah atau memanipulasi exspresi predikat kalkulus di atas, dengan mengakses komponennya sendiri-sendiri dan menduga kalimat baru.
- Sebagai contoh : kita dapat menetapkan semua nilai dari X, dimana X: satu hari dan semnggu. Kalimatnya menjadi sebagai berikut:

  Cuaca (X, hujan)

2.4. ***First Order Predicate Calculus***

Menggunakan *Quantifier :*

- *Universal Quantification* (∀x) ∀: *all, every*, untuk semua
- *Existential Quantification* (∃x) ∃ :*some, at least one*, terdapat, beberapa

Contoh:

Everybody loves Mary $\rightarrow$ $(\forall x)$ [loves (x,Mary)]

Somebody loves Mary $\rightarrow$ $(\exists x)$ [loves (x, Mary)]
All swans are white $\rightarrow$ $(\forall x)$ [swan (x) $\rightarrow$colour (x, white)]
There is ablack swan $\rightarrow$ $(\exists x)$ [Swan (x) $\wedge$ colour (x, black)]

There is Somebody who loves every one :

$(\exists x)$ $(\forall y)$ [loves (x,y)]

Everybody is loved by some one :

$(\forall x)$ $(\exists y)$ [loves (x,y)]

Table persamaan logika tambahan untuk predicate calculus

Refereces	**Persamaan Logika**
P_{12}	$\sim(\exists_x)$ [P(x)] $\equiv$ $(\forall_x)$ [$\sim$ P(x)]
P_{13}	$\sim(\forall_x)$ [P(x)] $\equiv$ $(\exists_x)$ [$\sim$ P(x)]
P_{14}	$(\forall_x)$ [P(x) $\wedge$ Q(x)] $\equiv$ $(\forall_x)$ [P(x)]$\wedge$ $(\forall y)$ [Q(y)]
P_{15}	$(\exists_x)$ [P(x) $\vee$ Q(x)] $\equiv$ $(\exists_x)$ [P(x)]$\wedge$ $(\exists y)$ [Q(y)]
P_{16}	$(\forall_x)$ [P(x)] $\equiv$ $(\forall_x)$ [P(y)]
P_{17}	$(\exists_x)$ [P(x)] $\equiv$ $(\exists_x)$ [P(y)]

Clause formation

Merupakan kreasi clauses dari beberapa Wff (*Well formed formula*) *prosedure* mengikuti 9 langkah yang semestinya dilakukan berurutan, ketika *aplicable*

1. Eliminasi Simbol implikasi (Implication / $\rightarrow$) semua simbol implikasi di hilangkan dengan bersamaan logika $P_{2.}$ ($\sim$ P v Q $\equiv$ P $\rightarrow$ Q)
2. *Reduce Scope of the negation Symbol* : jadi tanda negasi dipindahkan kedalam tanda kurung dekat pada setiap atom formula yang di pakai. (gunakan P_1, P_3, P_4, P_{12} & P_{13})
3. Standarisasi variabel : Proses ini menamakan ulang setiap variabel juga setiap Quantifier mempunyai variabel sendiri. Ini menjamin variabel-variabel dihubungkan pada Quantifier mereka secara bebas dari bentuk wff. (gunakan P_{16} & P_{17})
4. Eliminasi *Existential Quantifier* $(\exists x)$: Ini menyediakan penggunaan fungsi (function)skolem.

   Sebuah fungsi skolem merupakan fungsi pemetaan tambahan yang

menggantikan sebuah variabel melewati *existential Quantifiers.*

Contoh: $(\forall x)\ (\exists y)$ [loves (x,y)]

Sekarang sebuah fungsi g dapat di definisikan bahwa $y = g(x)$ dan setiap x akan secara otomatis dipetakan dalam beberapa y yang ada menurut formula (pilih y), maka:

$(\forall x)$ [loves (x, g(x))]

5. *Convert to Prenex Form:*

   Ini merupakan perpindahan secara sederhana dari semua universal Quantifier ke depan dari wff. Sejak setiap Quantifier mempunyai Variabelnya sendiri, *correct stops*, tidak hilang.

6. Letakkan badan utama dari wff kedalam bentuk *conjuctive* normal : jadi sekumpulan clauses di produksi. (Gunakan P_6)

7. Eliminasi *the universal Quantifier* $(\forall x)$

   Ini merupakan proses sederhana menghilangkan mereka dari depan wff. Asumsi ditetapkan bahwa semua unbound variabel adalah universally Quantifier, jadi tidak memerlukan secara exsplisit kedudukan ini.

8. Eliminasi semua simbol $\wedge$:

   Setiap *clause* dipisahkan sekarang. Sejak tafsiran selalu dikonsentrasikan dengan *conjuction* dari dasar pikiran , <u>*clauses*</u> boleh dipisahkan.Ini merupakan notasi.

9. Standarisasi sebagai variabel :

   Ini merupakan proses penamaan kembali setiap variabel, jadi tidak ada variabel lebih dari satu anak kalimat.

   Contoh : *USE Algorithm Convert to clauses Form* :

   $(\forall x)\ \{ P(x) \rightarrow \{ (\forall y)\ [\ P(y)\ \rightarrow P(f9x,y))] \wedge\ \sim (\forall y)\ [Q(x,y) \rightarrow P(y)]\}\}$

Langkah 1.

Menjadi:

$(\forall x)\ \{\sim P(X) \vee \{(\forall y)\ [\ \sim P(y)\ V\ P(f(x,y))] \wedge \sim (\forall y)\ [\ \sim Q(x,y) \vee P(y)]\}\}$

Langkah 2.

$(\forall x)\ \{\sim P(x) \vee \{(\forall y)\ [\ \sim P(y)\ V\ P(f(x,y))] \wedge (\exists y)\ [Q(x,y) \wedge \sim P(y)]\}\}$

Langkah 3.

$(\forall x)\ \{\sim P(x) \vee \{(\forall y)\ [\sim P(y)\ V\ P\ (f\ (x,y))] \wedge (\exists Z)\ [Q(x,Z) \wedge \sim P\ (Z)]\}\}$

Langkah 4.

$(\forall x)\ \{\sim P(x) \vee \{(\forall y)\ [\sim P(y)\ V\ P\ (f\ (x,y))] \wedge [Q(x,g(x)) \wedge \sim P\ (g(x))]\}\}$

Langkah 5.

$(\forall x)\ (\forall y)\ \{\sim P(x)\ V\ \{[\sim P\ (y) \vee P(f\ (x,y))] \wedge [Q(x,g(x\ )) \wedge \sim P\ (g(x))]\}\}$

Langkah 6.

$(\forall x)\ (\forall y)\ \{[\sim P(x) \vee \sim P(y) \vee P(f\ (x,y))] \wedge [\sim P(x) \vee Qn(x,g(x))]$

$\wedge [\sim P\ (x) \vee \sim P(g(x))]\}$

Langkah 7.

$\{[\sim P(x) \vee \sim P(y) \vee P\ (f(x,y))] \wedge [\sim P(x) \vee Q(x,g(x))] \wedge [\sim P(x) \vee \sim P\ (g(x))]\}$

Langkah 8.

$\sim P(x) \vee \sim P(y) \vee P\ (f(x,y))$.

$\sim P(x) \vee Q\ (x,g(x))$

$\sim P(x) \vee\ P(g(x))$.

Langkah 9.

$\sim P(x_1) \vee \sim\ P\ (y)\ \vee P\ (\ f(x_1,y))$.

$\sim P(x_2) \vee Qn\ (x_2,g\ (x_2))$.

$\sim P(x_3) \vee \sim\ P\ (g\ (x_3))$.

<u>Contoh yang lain:</u>

1). All Romans who knows Marcus either hate caesar or think that anyone who hates any one is crazy.

Kita dapat merepresentasikan kalimat diatas kedalam wff sebagai berikut:

$(\forall_x)\ [\ Roman\ (x)\ \wedge\ know\ (x,Marcus\ )] \rightarrow [hate\ (x\ ,Caesar\ )\ v\ (\forall_y)\ (\exists_Z\ )\ hate(y,z) \rightarrow thinkcrazy\ (x,y))]$.

2). There is some one who's going to pay for all the service There for each of the service is going to be paid for by some one.

Bentuk predicate calculus kalimat diatas sebagai berikut :

P(x) = x is a person
S(y) = y is a service
G (x,y) = x is going to pay for y

Kita lihat bahwa jika dasar pikiran atau alasan :

$(\exists_x)$ [P(x) $\wedge$ $(\forall_y)$ [S (y) $\rightarrow$ G (x,y)]]

maka kesimpulan :

$(\exists_y)$ [S(y) $\rightarrow$ $(\exists_x)$ [P(x) $\wedge$ G (x,y)]]

maka wff:

$(\exists_x)$ [P(x) $\wedge$ $(\forall_y)$ [S (y) $\rightarrow$ G (x,y)]] $\rightarrow$ $(\exists_y)$ [S(y) $\rightarrow$ $(\exists_x)$ [P(x) $\wedge$ G (x,y)]]

Representasi kenyataan-kenyataan (*facts*) sederhana dalam *logic*

Penggunaan dari *propositional logic* sebagai langkah / cara merepresentasikan dari pengetahuan dunia yang diperlukan sebuah sistem AI. *Propositional logic* adalah menarik sebab dia sederhana untuk menghadapi prosedure keputusan untuk keberadaanya.

Kita dapat dengan mudah menampilkan *fact-fact* dunia nyata sebagai proposisi logika yang ditulis sebagai wffs. Beberapa *facts* sederhana dalam *propositional logic* :

It is raining
Raining

It is sunny
Sunny

It is windy
Windy

If is raining, then is not sunny
Raining $\rightarrow$ $\neg$ sunny

Contoh penggunaan *predicate logic* sebagai cara untuk merepresentasikan pengetahuan ;

1. Marcus was a man
   man (marcus)

2. Marcus was a pompeian
   pompeian (marcus)
3. All pompeian were Romans
   $\forall_X$: pompeian (x) $\rightarrow$ roman (x)

4. Caesar was a ruler
   ruler (caesar)

5. All romans were either loyal to caesar or hated him
   $\forall_X$: roman (x) $\rightarrow$ loyal to (x, caesar) v hate (x ,caesar)

   Inclusive Interpretation
   $\forall_X$: roman(x) $\rightarrow$ [(loyal to (x, caesar) v hate(x ,caesar) $\wedge$ $\neg$(loyalto(x, caesar) $\wedge$ hate(x,caesar))]

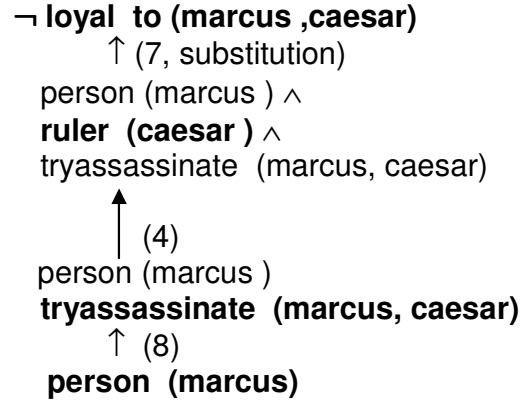
6. Every one is loyal to some one
   $\forall_X$: ($\exists_y$) : loyal (x,y)

   some one to whom every one is loyal
   $\exists_x$:$\forall_y$: loyal to (x,y)

7. People only try to assassinate rulers they are not loyal to
   $\forall_X$: :$\forall_y$: person(x) $\wedge$ rulers (y) $\wedge$ tryassassinate (x,y) $\rightarrow$ $\neg$ loyal to(x,y)

8. Marcus tried to assassinate Caesar
   tryassassinate (marcus, Caesar)

9. All men are people
   $\forall_X$: man (x) $\rightarrow$ person (x)



Gambar 2.1. memperlihatkan sebuah usaha membuktikan bahwa marcus tidak loyal (setia) kepada caesar:

¬ loyal to (marcus , caesar)

Bila pengetahuan di atas dan untuk membuktikan bahwa marcus tidak loyal

kepada caesar dituangkan dalam bahasa Turbo Prolog, sebagai berikut:

```
Domains
        Simbol = string

Predicates
        man(Simbol)
        ruler(Simbol)
        tryassassinate(Simbol, Simbol)
        person(Simbol)
        notloyalto(Simbol., Simbol )

Clauses
        man(marcus).
        ruler(caesar).
        tryassassinate(marcus, caesar).
        person(X)      :-      man(X).
        notloyalto(X, Y) :-    person(X),
                               ruler(Y),
                               tryassassinate(X, Y).
```

setelah itu tekan tombol ALT + R untuk RUN , sehingga muncul menu dialogue sebagai berikut:

```
Goal:  notloyalto(A,B).
       A=marcus;
       B=caesar;
       No
Goal:  notloyalto(P, caesar).
       P=marcus;
       No
```