Sistem Pakar Pencarian Silsilah Keluarga Menggunakan Pemrograman Prolog

Dwi Retnoningsih
Program Studi Teknik Informatika, Universitas Sahid Surakarta
Jl. Adi Sucipto 154, Jajar, Surakarta, 57144, Telp. (0271) 743493, 743494
Email: dw1retno@yahoo.co.id

Abstract

Family tree search process for someone who is a member of the family would quickly be known. But instead for someone other than a family member would be a difficult thing to do if you do not know with certainty the origin of the family. The purpose of this study is making an expert system application to search the family tree using Prolog Language.

The research method that I use is the method of back tracking. This research resulted in an expert system for searching the family tree.

Keywords: *back tracking, Depth-First Search, Family Tree, prolog,Searching,*

Pendahuluan

Latar Belakang

Pencarian sil-silah keluarga menjadi tema yang menarik bagi penulis untuk diteliti. Karena membutuhkan kecermatan dalam penentuan metode dan alur logika yang logis sehingga mudah dalam merunut dari induk ke anak yang paling terdalam.

Berdasarkan uraian tersebut, penulis mencoba membuat sebuah aplikasi sistem pakar untuk pencarian sil-silah keluarga dengan cara yang sederhana dan mudah untuk dipahami.

Permasalahan

Proses pencarian sil-silah keluarga bagi seseorang yang menjadi anggota dari keluarga tersebut tentu akan cepat dapat diketahui. Tetapi sebaliknya bagi orang lain yang bukan menjadi anggota keluarganya tentu menjadi suatu hal yang sulit dilakukan jika belum mengetahui dengan pasti asal-usul keluarga tersebut.

Tujuan Penulisan

Membuat suatu aplikasi sistem pakar untuk pencarian sil-silah keluarga menggunakan Pemrograman Prolog.

Landasan Teori

Pemrograman Prolog

Sebelum Turbo Prolog keluar telah ada beberapa Aplikasi Pemrograman Prolog yang lain di pasaran, antara lain Prolog-86 Plus, A.D.A Prolog, Mprolog dan lainnya. Turbo Prolog menggabungkan kemampuan Prolog tradisional dengan kecepatan dan keampuhan bahasa lainnya.

Prolog (*Programming in Logic*) dibangun atas dasar pemrograman alamiah dan logika. Prolog merupakan bahasa *deklaratif*, artinya jika kita memberikan fakta dan aturan, prolog akan menyelesaiakan secara deduktif, atau dari banyak fakta dan aturan kemudian diturunkan kesimpulan sebagai jawaban. Bukti bahwa prolog adalah bahasa deklaratif adalah **X adalah leluhur Z.** Maka harus didefinisikan terlebih dahulu logikanya (mis. dalam bahasa Indonesia) : X adalah leluhur Z jika x orang tua Z, atau X adalah leluhur Z jika x orang tua B dan B leluhur Z. Aturan tersebut dalam pemrograman prolog ditulis sebagai aturan rekursif sebagai berikut :

Leluhur(X,Z) if orangtua(X,Z).

Leluhur(X,Z) if orangtua(X,B) and leluhur(B,Z).

Hanya dengan menuliskan aturan tersebut maka akan dapat diketahui siapa leluhur Z, siapa keturunan X, atau membuktikan kebenaran bahwa X leluhur Z. Pada pemrograman prolog cukup memberikan data hubungan (fakta) orang tua secukupnya seperti contoh tersebut, tanpa harus merinci cara mencari jawaban atau cara membuktikannya.

Struktur sintak pada Pemrograman Turbo Prolog terdiri dari beberapa bagian pokok dan menggunakan deklarasi berikut **Deklarasi** (jumlah deklarasi dalam program), ***Domains*** (tanpa, satu atau lebih), ***Predicats*** (satu atau lebih), ***Clauses*** (tanpa, satu atau lebih), ***Data Base*** (tanpa, satu atau lebih), ***Goal*** (tanpa atau satu).

Selain itu terdapat dua jenis variabel dalam turbo prolog yaitu Variabel Bebas dan Variabel Tak Bebas. Variabel Bebas adalah variabel yang belum diketahui isinya oleh Turbo Prolog, sedangkan Variabel Tak Bebas adalah variabel yang sudah diberikan isinya. Penulisan nama variabel bebas dimulai dengan huruf pertama besar dan variabel tak bebas dimulai dengan huruf kecil.

Domains

Pada bagian inilah program Turbo Prolog mendeklarasikan besaran. Besaran yang akan dipakai adalah Variabel di dalam *Predicat*.

Predicats

Digunakan untuk mendeklarasikan kata kerja yang akan dipergunakan dalam *GOAL*. Bentuk umumnya adalah PREDICATS (Domain 1, Domain 2, …). Tetapi *predicats* tidak harus mempunyai parameter. Biasanya *predicats* ini digunakan untuk memberi nama suatu sub program. Dengan memanggil nama *predicats*-nya saja maka sub program akan dilaksanakan.

Clauses

Clauses adalah tempat pemberian nilai dan tempat pembuatan *Rule*. Cara penulisan *Clauses* dapat disederhanakan menjadi; untuk IF dapat ditulis sebagai IF atau ":" , untuk AND dapat ditulis AND atau tanda Koma (,), untuk OR ditulis dengan tanda (;)

Data Base

Data Base adalah tempat untuk menyatakan fakta dalam program yang merupakan bagian dari Basis data Dinamik (berubah).

Goal

Goal dapat diartikan tujuan yang ingin dicapai. Macam-macam penggunaan *Goal* ada dua yaitu *Goal* Eksternal dan *Goal* Internal. Goal eksternal diketikkan langsung pada *Promp "GOAL : "* di jendela dialog pada kompiler terpadu Turbo

Prolog. Cara ini hanya bisa menjalankan program dari lingkungan kompiler Turbo Prolog. Agar dapat dijalankan dilingkungan DOS, harus menggunakan *Goal* Internal yang disimpan dalam bagian program *Goal* yang dituliskan pada bagian program.

Pokok perbedaan Prolog dengan bahasa lain adalah karena bersifat deskriptif atau deklaratif, sedang bahasa pemrograman lainnya ada yang bersifat prosedural atau imperatif. Artinya Turbo Prolog hanya membutuhkan deklarasi atau uraian masalah, sedangkan yang lainnya memerlukan perintah. Sebagai bukti bahwa Turbo Prolog merupakan bahasa deklaratif adalah jika ingin menerangkan bahwa A adalah orang tua B, maka dalam Pemrograman Prolog cukup ditulis **Orangtua(A,B).**

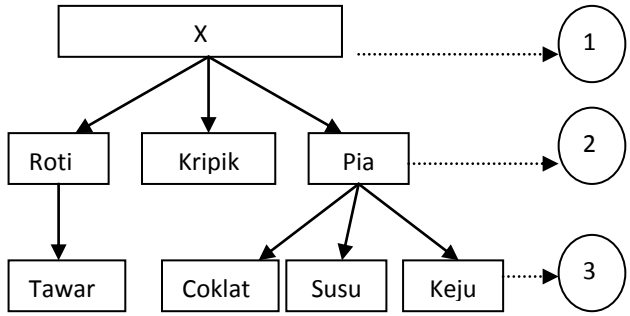
Lacak Balik (*Back Tracking)*

Implementasi program aplikasi untuk pencarian sil-silah keluarga sebenarnya banyak metode yang dapat digunakan. Tetapi cukup rumit dan sulit untuk dipahami bagi orang awam yang mungkin tidak familiar dengan pemrograman prolog.

Pada penelitian ini penulis menggunakan metode lacak balik atau lebih dikenal dengan *back tracking* adalah suatu metode dalam pencarian jawaban dan pemenuhan goal. Pada pemecahan suatu persoalan, misalnya menyederhanakan persamaan matematika, pertama dicoba menempuh suatu jalan. Jika akhirnya gagal, tentu akan menempuh jalan lain. Cara ini dipakai juga dalam Prolog yang dikenal dengan Lacak Balik / *Back Tracking*. Pada proses lacak balik ini Turbo Prolog menggunakan kiat pencarian ke dalam pertama (*Depth-First Search*). Contoh program 1 berikut merupakan contoh program yang menggunakan metode ini:

```
Traces
Domains
    Orang, Makanan, Rasa = symbol,
Predicats
    Suka(orang,makanan),
    Berasa(makanan,rasa),
    Makanan(makanan),
Clauses
    Suka(didit,X) : -makanan(X),
       berasa(X,keju),
    Berasa(roti,tawar),
    Berasa(pia,coklat),
    Berasa(pia,keju),
    Makanan(roti),
    Makanan(kripik),
    Makanan(pia),
```

Agar lebih mempermudah dalam memahami metode *back tracking*, Gambar 1 berikut merupakan gambar bagan untuk mengetahui alur logika secara sederhana dari metode *back tracking*.



Gambar 1 Alur Logika Secara Sederhana Metode *Back Tracking*

Keterangan dari Gambar 1 tersebut adalah 1 = goal yang ingin dicapai, 2 = makanan : roti, kripik, pia, 3 = berasa : tawar, coklat, susu dan keju. Jika dibuat goal sebagai berikut: Suka(didit,apa), pada saat ini unifikasi antara pertanyaan (*goal*) **Suka(didit,apa)** dengan aturan **Suka(didit,X)**. Variabel **Apa** berunifikasi dengan **X** yang keduanya masih bebas.

Aturan tersebut terdiri dari *sub goal* : makanan dan berasa. Karena *sub goal* harus dipenuhi, maka Prolog mulai memanggil *sub goal* paling atas hingga paling bawah. Artinya pertama Prolog melakukan unifikasi ***sub goal*** **makanan(X)** dengan kumpulan *clausa* makanan yang teratas yaitu makanan(roti). Unifikasi berhasil menyebabkan X terikat dengan roti.

Pada contoh tersebut Prolog mencatat titik lacak balik dengan klausa makanan karena ada klausa makanan yang lain (makanan(kripik)). Prolog kembali / *return* dengan berhasil (X=roti) sehingga Prolog melanjutkan tugasnya ke sub goal berikutnya yaitu **berasa(X,Keju)** yang ditafsirkan sebagai **berasa(roti,keju)** karena X sudah diikat roti.

Pemanggilan ini dimaksudkan untuk memeriksa kebenaran bahwa **Didit suka roti yang berasa keju**. Tetapi karena ada fakta yang memperlihatkan roti yang berasa keju, maka Prolog menyimpulkan bahwa Didit suka roti adalah anggapan yang salah meskipun dalam kenyataannya benar. Gejala ini disebut dengan *Closed-World Assumption.*

Sehingga dalam hal ini Prolog mengalami kegagalan untuk memperoleh jawaban. Kegagalan terjadi pada sub goal **berasa**. Prolog mengerti untuk mencari jawaban dari sub goal terdekat yaitu makanan. Artinya Prolog harus mengganti anggapan lain tentang makanan kesukaan Didit. Pada pemanggilan sub goal makanan yang kedua kalinya, Prolog memeriksa catatan titik lacak balik. Disini ada titik lacak balik dari clausa makanan(kripik). Maka proses dilaksanakan seperti diatas dan Prolog menyimpulkan bahwa Didit suka kripik adalah anggapan yang salah. Begitu selanjutnya sehingga Prolog mencatat lacak titik balik yang lainnya yaitu makanan(pia) dengan instantiasi X=pia, Prolog melanjutkan ke subgoal **berasa(pia,keju)** dan berhasil karena ada fakta tersebut, sehingga Prolog menjawab berikut :

```
Apa = Pia
1 solution
```

Metode *back tracking* dalam implementasi pemrograman prolog dapat dipadukan dengan menggunakan metode pencarian jawaban I, metode pencarian jawaban II, atau metode pencarian jawaban III.

Metode metode pencarian jawaban I merupakan cara pengendalian program sehingga tidak terjadi lacak balik yang tidak perlu. Mekanisme lacak balik pada Prolog dapat menimbulkan pencarian yang tidak perlu, sehingga jalannya program tidak effsien. Pada metode ini menggunakan Predikat *Fail* dan *True*.

Bisa terjadi Prolog melakukan lacak balik meskipun *goal* sudah tercapai dan tidak perlu melakukan apapun. Sehingga perlu diketahui cara pengendalian program supaya tidak terjadi lacak balik yang tidak perlu.

Seperti kita ketahui lacak balik terjadi apabila menemui kegagalan. Pada keadaan tertentu perlu memaksa lacak balik untuk memperoleh alternatif jawaban yang lain. Memanggil Predikat *Fail* sama seperti membandingkan 1⇔2. Contoh program 2:

```
Predicates
    Ibu(symbol,symbol),
    Seseorang,
Clauses
    Ibu(acih,uup),
    Ibu(enah,bek),
    Ibu(ani,asep),
    Seseorang : - ibu(X,Y),
                Write( X, " adalah ibu ",Y," "),
    Fail
Goal
    seseorang
```

Jika program dijalankan akan menjawab sebagai berikut :

- Jika memakai goal eksternal Ibu(X,Y) akan menyebutkan semua fakta yang ada sebagai berikut X = Acih, Y = Uup, X = Enah, Y =Bek, X = Nani, Y = Asep
- Jika dengan goal internal seperti tampak dalam program dan tidak ada predikat *fail*, maka Prolog hanya memberi satu jawaban Acih adalah ibu Uup.

Tetapi karena ada predikat *fail* setelah menuliskan jawaban diatas, Prolog dipaksa melakukan lacak titik balik ke subgoal sebelumnya, yaitu : Ibu(X,Y). Kemudian karena predikat *Write* tidak mempunyai alternatif lain, Prolog gagal melakukan unifikasi dengan ibu yang kedua yaitu Ibu(Enah,Bek), maka Prolog menuliskan Enah adalah ibu Bek. Proses berulang sampai tidak ada alternatif jawaban lain. Pada saat inipun Prolog tetap gagal. Karena tidak ada jawaban lain, Prolog terpaksa berhenti dalam keadaan gagal, karena adanya predikat *Fail*.

Tetapi karena pada program memberikan *Clausa* Seseorang berupa fakta / *rule* tanpa suatu nilai, maka program berhenti dalam keadaan berhasil. Karena sekarang titik lacak baliknya adalah seseorang yang tanpa harus memenuhi syarat. Pemanggilan *Clausa* yang mengandung lebih dari satu jawaban disebut *Non deterministic*, sedangkan *clausa* yang sebaliknya disebut *Deterministic*. Dari goal tersebut jelas bahwa Prolog tidak akan pernah memanggil klausa setelah predikat *fail*, jadi tidak ada artinya meletakkan sesuatu setelah predikat *fail*.

Metode pencarian jawaban II merupakan cara pengendalian program sehingga tidak terjadi lacak balik yang tidak perlu. Pada metode ini juga menggunakan Predikat *Fail, True* serta *Cut.*

Cut berarti memotong jejak untuk melakukan lacak balik. Predikat ini ditandai dengan tanda seru (**!**). *Cut* berfungsi seperti pintu perangkap tikus, begitu ada yang melewati pintu perangkap, dia hanya akan dapat berputar didalam perangkap. Contoh cara untuk menggunakan Predikat *cut* adalah sebagai berikut :

X : - A, B, !, C, D,
X : - E

Setelah melakukan unifikasi dengan aturan X yang pertama, Prolog mencari titik lacak balik karena ada X yang lain. Jika gagal, lacak balik ke klausa X berikutnya. Jika berhasil melanjutkan ke B. apabila gagal, lacak balik ke X berikutnya, jika A tidak mengandung titik lacak balik. Tetapi jika B berhasil, Prolog memasuki Cut sehingga titik lacak balik sebelumnya terhapus semua. Akibatnya apabila kemudian gagal, tidak terjadi lacak balik ke B, A atau tidak juga ke X berikutnya. Setelah memasuki Cut apabila C gagal, program berhenti dengan gagal, tetapi jika C berhasil proses berlanjut ke D, dan apabila D gagal lacak balik hanya ke C. jika D berhasil program selesai dalam keadaan berhasil.

Prediksi *cut* terbagi dalam dua bagian, yaitu *Cut* hijau **/ *Green Cut.*** Kehadiran dan ketiadaanya tidak mengubah logika program, tapi diperlukan untuk meningkatkan efisiensi / kecepatan. Sedangkan *Cut* Merah / *Red Cut* untuk mencegah perhatian Prolog terhadap alternatif subgoal. Contoh program3:

```
Trace
Predicates
      Minuman(symbol,symbol)
      Tes_baik(symbol)
      Minuman_baik
Clauses
      Minuman(susu,sehat)
     Minuman(kopi,menyegarkan)
     Minuman(arak,berbahaya)
      Minuman(sirup,berenergi)
      Minuman_baik : - minuman(jenis,sifat)
                  Tes_baik(sifat)
                  Write(jenis," minuman ", sifat, " ")
                  Fail
     Minuman_baik
     Tes_baik(berbahaya) : - !,  fail
     Tes_baik(_)
Goal
     Minuman_baik
```

Metode pencarian jawaban III adalah cara pengendalian program sehingga tidak terjadi lacak balik yang tidak perlu. Metode ini masih menggunakan predikat *Fail* dan *True*. Kadang juga menggunakan predikat *Not.*

Predikat *Not* berfungsi untuk membalik keadaan. Menggunakan predikat *Not* maka pemanggilan yang berhasil menjadi gagal dan sebaliknya. Penggunaan Predikat

Not khususnya untuk atom atau mengandung variabel tapi harus sudah terikat dan *Not* tidak boleh dipakai di kepala klausa seperti :

Not(Aturan1,(…)) : - sel,… (salah)
Not(Besar(X)),… (benar)

Sub klausa Besar diatas digunakan untuk memeriksa apakah sesuai berukuran Besar atau tidak.

Metode Penelitian

Pada penelitian Sistem Pakar Pencarian Silsilah Keluarga ini penulis mengunakan metode lacak balik atau lebih dikenal dengan *back tracking.* Metode *back tracking* adalah suatu metode dalam pencarian jawaban dan pemenuhan goal.

Hasil dan Pembahasan

Pada kasus yang pertama ini contoh sebuah program sederhana tentang hubungan keluarga. Program 1 merupakan contoh program sederhana tentang sebuah hubungan keluarga (Gambar 2) .

```
/* Prolog Keluarga */
Domains
        Orang = symbol
Predicates
        Orangtua(orang,orang)
        Perempuan(orang)
        Lakilaki(orang)
Clauses
        Orangtua(tono,bob).
        Orangtua(aminah,nisa).
        Perempuan(aminah).
        Lakilaki(tono).
        Lakilaki(bob).
```

Gambar 2. Program 1

Cek apakah Tono adalah orang tua dari Bob?. Maka prolog akan menjawab Yes karena pada *Clauses* sudah dideklarasikan *statement*-nya.

```
Goal: Orangtua(tono,bob)
Yes
```

Siapakah orang tua Nisa? Sehingga tentukan goalnya adalah Orangtua(y,nisa) dan prolog menjawab Aminah. Artinya proses pencarian pada statemen prolog ditemukan jawabanya.

```
Goal: Orangtua(Y,nisa)
Y=aminah
1 Solution
```

Sebuah keluarga umumnya terdiri dari bapak (suami), ibu (istri), dan anak. Pada kasus yang kedua ini penulis mengembangkan program 1 menjadi lebih lengkap dalam satu keluarga. Bagaimanakah prolog mendefinisikan fakta ini?. Caranya yaitu pada *predicats* tinggal ditambahkan kata kerja yaitu bapak, ibu, anak, suami, istri yang akan dipergunakan dalam menentukan *GOAL*. Pengembangannya pada program 2 dapat di lihat pada Gambar 3.

```
Predicates
        Umur(orang,um)
        Suami(orang,orang)
        Istri(orang,orang)
        Anak(orang,orang)
        Ibu(orang,orang)
        Bapak(orang,orang)
```

Gambar 3. Program 2

Kemudian pada *Cluses* ditambahkan *rule* untuk suami dan istri yaitu :

Istri(x,y) if Suami(x,y).

Rule ini akan dipergunakan dalam menentukan pencarian istri dan suami.

```
Istri(X,Y) if Suami(X,Y).
```

Jika dalam suatu keluarga tersebut mempunyai anak bagaimanakah *rule*nya?. *Rule*nya yaitu pada *Cluses* ditambahkan *rule* berikut:

Ibu(a,i) : Anak(i,a)

Bapak(c,b) : Anak(B,D), Suami(C,D)

```
Ibu(A,I):-anak(I,A).
Bapak(C,B):-anak(B,D), suami(C,D).
```

Jika ditemukan kenyataanya seorang suami dan istri memiliki anak lebih dari satu. Sehingga ada fakta adik dan kakak. Bagaimanakah prolog mendeklarasikan *rule*nya?. Caranya adalah pada *Clauses* ditambahkan *statement* berikut :

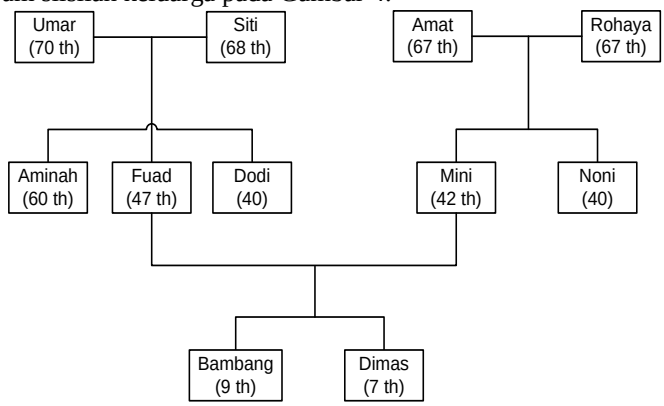
Kakak(x,y) yang berarti x adalah kakak y.

dan **Adik(x,y)** yang berarti x adalah adik y.

```
Kakak(orang,orang)
Adik(orang,orang)
```

Agar fakta mendekati kebenaran maka pada program prolog dapat ditambahkan umur dari masing-masing orang.

Pada kasus yang ketiga, penulis mengembangkan lagi program prolog silsilah keluarga ini untuk dapat mengetahui kakek, nenek, pakdhe, budhe, om, dan tante. Agar lebih mudah dalam memahami silsilah keluarga pada contoh kasus ini penulis membuat gambar diagram silsilah keluarga pada Gambar 4.



Gambar 4. Diagram Pohon Keluarga

Pada Gambar 4 merupakan gambar diagram pohon keluarga. Penjelasannya adalah terdapat sebuah keluarga yang pertama yaitu Umar (70 tahun) mempunyai istri bernama Siti (68 tahun) mempunyai tiga orang anak yaitu Aminah (60 tahun), Fuad (47 tahun), Dodi (40 tahun). Pada keluarga yang kedua Amat (67 tahun) mempunyai istri bernama Rohaya (67 tahun) mempunyai dua orang anak yaitu Mini (42 tahun) dan Noni (40 tahun). Ternyata salah satu anak dari Umar dan Siti yang bernama Fuad merupakan suami dari Mini yang merupakan anak dari Amat dan Rohaya. Fuad dan Mini mempunyai anak bernama Bambang (9 tahun) dan Dimas (7 tahun).

Bagaimanakah prolog mendefinisikan statementnya?. Dari contoh program 1 dan program 2 tinggal digabungkan dan ditambahkan beberapa fakta sesuai dengan gambar diagram pohon keluarga pada Gambar 4.

Melihat dari diagram pohon keluarga pada gambar 4 terlihat fakta bahwa Umar dan Amat adalah kakek dari Bambang dan Dimas. Serta Siti dan Rohaya adalah nenek Bambang dan Dimas. Dari fakta tersebut perlu penyesuaian kata kerja kakek dan nenek pada *predicate*nya.

Setelah ditelusuri lebih lanjut, menimbulkan pertanyaan lain yaitu bukankah Aminah adalah budhe dari Bambang dan Dimas?, Dodi adalah Om dari Bambang dan Dimas? Serta Noni adalah tante dari Bambang dan Dimas?. Bagaimanakah prolog mendefinisikan fakta-fakta tersebut?. Caranya yaitu pada *Predicate* ditambahkan kata kerja budhe, om, dan tante.

```
Predicates
        Umur(orang,um)
        Pria(orang)
        Wanita(orang)
        Suami(orang,orang)
        Istri(orang,orang)
        Anak(orang,orang)
        Ibu(orang,orang)
        Bapak(orang,orang)
        Kakak(orang,orang)
        Adik(orang,orang)
        Budhe(orang,orang)
        Om(orang,orang)
        Tante(orang,orang)
        Kakek(orang,orang)
        Nenek(orang,orang)
```

Gambar 5. Pengembangan Program 1

Sebelumnya yang perlu dipastikan bahwa istri, nenek dan tante berjenis kelamin wanita, sedangkan suami, kakek dan om berjenis kelamin pria. Hal ini penting untuk didefinisikan terlebih dahulu karena digunakan untuk membedakan sebutan antara suami dan istri, kakek dan nenek, om dan tante. Sehingga program harus dikembangkan lagi dan pada *Clauses* dilengkapi statemen berikut :

```
Ibu(A,I):-anak(I,A), wanita(A).
Bapak(C,B):-anak(B,D), suami(C,D), pria(C).

Kakek(X,K):-bapak(X,B), bapak(B,K).
Kakek(X,K):-bapak(X,B), ibu(B,K).

Nenek(X,N):-ibu(X,B), ibu(B,N).
Nenek(X,N):-ibu(X,B), bapak(B,N).
```

Gambar 6. Pengembangan Program 2

Sekarang saatnya untuk melakukan pelacakan silsilah keluarga tersebut, dengan mencocokan diagram pohon keluarga pada Gambar 4.

Misalnya untuk mengetahui **siapakah suami dari Siti?.** Maka untuk menentukan tujuan tersebut pada *dialog window* cukup memberikan perintah **Suami(Y,Siti)**. Jawaban dari prolog adalah **Umar**.

```
Goal: Suami(Y,siti)
Y=umar
1 Solution
```

Mari mencoba menanyakan pasangan masing-masing suami dan istri?

```
Goal: Suami(X,Y)
X=umar, Y=siti
X=amat, Y=rohaya
X=fuad, Y=mini
3 Solutions
```

Coba kita lacak apakah betul Mini adalah Rohaya?. Prolog menjawab Yes.

```
Goal: Anak(mini,rohaya)
Yes
```

Lacak apakah Amat adalah suaminya Siti? Prolog menjawab No, karena yang betul Amat adalah suami dari Rohaya.

```
Goal: Suami(amat,siti)
No
```

Tanyakan Mini anaknya siapa? Jawaban dari Prolog adalah Rohaya.

```
Goal: Anak(mini,X)
X=rohaya
1 Solution
```

Anaknya mini siapa saja? Yaitu Bambang dan Dimas.

```
Goal: Anak(Y,mini)
Y=bambang
Y=dimas
2 Solutions
```

Siapakah tantenya bambang?, Jawabanya adalah Noni.

```
Goal: Tante(X,bambang)
X=noni
1 Solution
Goal: _
```

Umar kakeknya siapa? Jawaban dari Prolog adalah Bambang dan Dimas.

```
Goal: Kakek(umar,Y)
Y=bambang
Y=dimas
2 Solutions
```

Rohaya neneknya siapa? Jawaban dari Prolog adalah Bambang dan Dimas.

```
Goal: Nenek(rohaya,X)
X=bambang
X=dimas
2 Solutions
```

Siapakah kakaknya Dodi?. Jawaban dari Prolog adalah Aminah dan Fuad.

```
Goal: Kakak(X,dodi)
X=aminah
X=fuad
2 Solutions
```

Siapakah adiknya Mini? Jawaban dari Prolog adalah Noni

```
Goal: Adik(X,mini)
X=noni
1 Solution
```

Dodi om-nya siapa? Prolog menjawab Bambang dan Dimas.

```
Goal: om(dodi,Y)
Y=bambang
Y=dimas
2 Solutions
```

Siapakah budhenya dimas? Prolog menjawab Aminah.

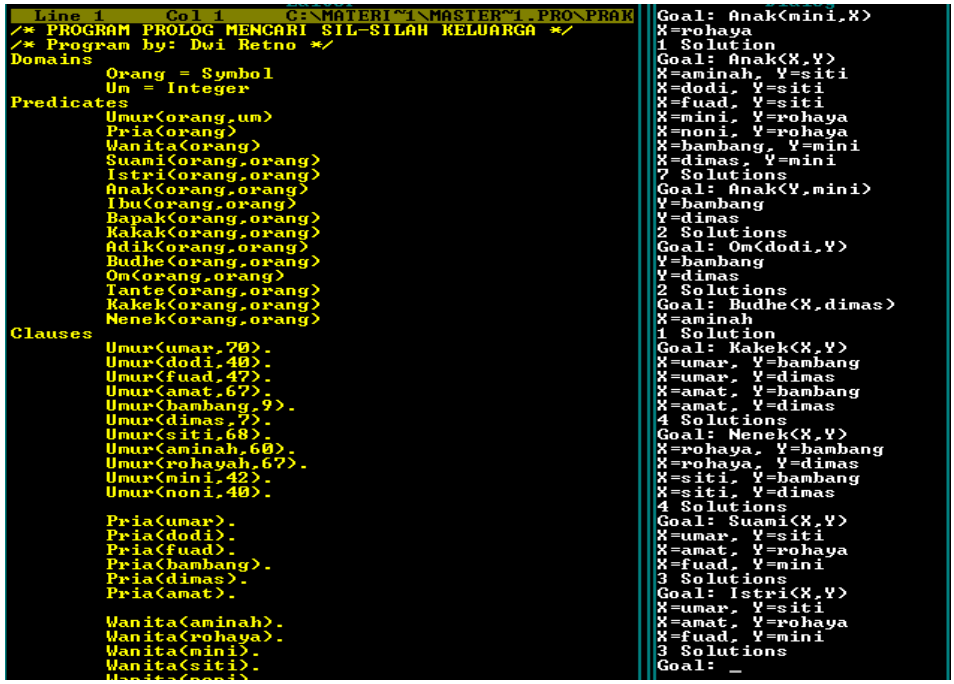
```
Goal: Budhe(X,dimas)
X=aminah
1 Solution
```

Gambar 7, Gambar 8, dan Gambar 9 berikut adalah hasil dialog antara user dengan program aplikasi sil-silah keluarga menggunakan pemrograman Prolog :

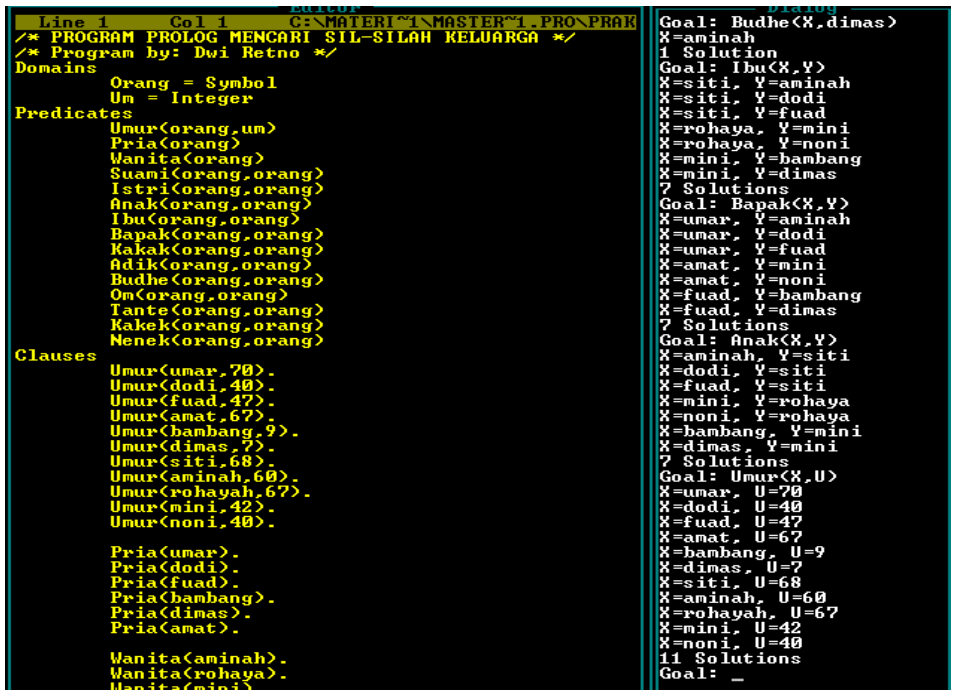
```
  Line 1      Col 1      C:\MATERI~1\MASTER~1.PRO\PRAK
/* PROGRAM PROLOG MENCARI SIL-SILAH KELUARGA */
/* Program by: Dwi Retno */
Domains
        Orang = Symbol
        Um = Integer
Predicates
        Umur(orang,um)
        Pria(orang)
        Wanita(orang)
        Suami(orang,orang)
        Istri(orang,orang)
        Anak(orang,orang)
        Ibu(orang,orang)
        Bapak(orang,orang)
        Kakak(orang,orang)
        Adik(orang,orang)
        Budhe(orang,orang)
        Om(orang,orang)
        Tante(orang,orang)
        Kakek(orang,orang)
        Nenek(orang,orang)
Clauses
        Umur(umar,70).
        Umur(dodi,40).
        Umur(fuad,47).
        Umur(amat,67).
        Umur(bambang,9).
        Umur(dimas,7).
        Umur(siti,68).
        Umur(aminah,60).
        Umur(rohayah,67).
        Umur(mini,42).
        Umur(noni,40).

        Pria(umar).
        Pria(dodi).
        Pria(fuad).
        Pria(bambang).
        Pria(dimas).
        Pria(amat).
```

```
Goal: Suami(Y,siti)
Y=umar
1 Solution
Goal: Anak(mini,rohaya)
Yes
Goal: Suami(X,Y)
X=umar, Y=siti
X=amat, Y=rohaya
X=fuad, Y=mini
3 Solutions
Goal: Suami(amat,siti)
No
Goal: Ibu(X,Y)
X=siti, Y=aminah
X=siti, Y=dodi
X=siti, Y=fuad
X=rohaya, Y=mini
X=rohaya, Y=noni
X=mini, Y=bambang
X=mini, Y=dimas
7 Solutions
Goal: Kakek(umar,Y)
Y=bambang
Y=dimas
2 Solutions
Goal: Nenek(rohaya,Y)
Y=bambang
Y=dimas
2 Solutions
Goal: Kakak(X,dodi)
X=aminah
X=fuad
2 Solutions
Goal: Adik(X,mini)
X=noni
1 Solution
Goal: Tante(X,bambang)
X=noni
1 Solution
Goal: _
```

Gambar 7. Dialog 1



Gambar 8. Dialog 2



Gambar 9. Dialog 3

Simpulan

Penelitian ini menghasilkan sebuah aplikasi yang dapat digunakan untuk proses pencarian sil-silah keluarga.

Daftar Pustaka

Arhami Muhammad.2006. *Konsep Dasar Sistem Pakar*. Yogyakarta : Andi

Kusrini. 2010. *Sistem Pakar Teori Dan Aplikasi*. Yogyakarta : Andi

Lukas Samuel, Anwar Toni, Panji Sony Wicaksono. 2006. *Simulasi Pencarian Jalan Menggunakan Algoritma Backtracking.* Jurnal Ilmiah Ilmu Komputer, Vol. 4 No. 1 Januari 2006, Universitas Pelita Harapan

Mega Pipin A.T. 2010. *Implementasi Algoritma Backtracking dengan Menggunakan Metode DFS (Depth First Search) Pada Penyelesaian Traveling Salesman Problem Suatu Digraph.* Skripsi. Universitas Negeri Malang

Yuwono Bambang. 2010. *Pengembangan Sistem Pakar Pada Perangkat Mobil Untuk Mendiagnosa Penyakit Gigi*. Proceeding Seminar Nasional Informatika 2010 (semnasIF 2010) ISSN: 1979-2328, UPN "Veteran" Yogyakarta