Implementasi *Section Facts* PROLOG sebagai Bahasa Pemograman *Artificial Intelligence* pada Proses Pemecahaan Masalah

Citra Kurniawan
Sekolah Tinggi Teknik Malang
Jl. Soekarno Hatta 94 Malang
airakurniawan@gmail.com

ABSTRAK

Pencarian data melalui metode *filtering* memungkinkan data yang diperoleh menjadi spesifik. *Section facst* pada prolog memungkinkan untuk mendeklarasikan fakta yang menjadi suatu *data base*. Pada penyelesaian masalah, *section facts* membuat data base berdasarkan sekumpulan fakta dan aturan. Fakta dan aturan didapatkan dari pengalaman dan pengetahuan yang telah didapatkan. Berdasarkan pengetahuan tersebut maka masalah dapat dipecahkan dengan cepat. Penggunaan metode *filtering* melalui *section facts* memungkinkan untuk mencari satu variabel yang berelasi dengan variabel yang lain. *Section Facts* merupakan salah satu komponen deklarasi dari bahasa prolog dalam *Artificial Intelligence*. Dalam penelitian ini didapatkan bahwa untuk menentukan *output* maka diperlukan sebuah deklarasi *fact* dan *rules* yang satu sama lain saling mempunyai relasi.

Kata kunci : *Artificial Intelligence*, Sec*tion Facts, Filtering, Data Base,* Prolog

PENDAHULUAN

Perkembangan bahasa pemograman memberikan metode dalam pemecahan masalah. Banyaknya parameter dalam masalah yang terjadi seringkali membuat proses pemecahan masalah menjadi semakin kompleks. Diperlukan sebuah metode yang dapat memberikan solusi secara cepat untuk memecahkan masalah. Salah satu teknologi yang dapat membantu untuk menyelesaikan masalah adalah *artificial intelligence*. *Artificial intelligence* (kecerdasan buatan) merupakan studi yang mempelajari kemampuan mesin (komputer) untuk menyelesaikan masalah dengan pemikiran dan kecedasan seperti manusia. Artificial intelligence membantu manusia untuk berkomunikasi dengan komputer dan menyelesaikan masalah berdasarkan pengalaman dan pengetahuan yang disimpan berupa data referensi (basis data) (Kurniawan, 2016).

Salah satu bahasa pemograman yang dipakai dalam *artificial intelligence* adalah Prolog. Prolog merupakan bahasa pemograman deklaratif (*object oriented*) yang dapat diaplikasikan untuk menyelesaikan masalah dengan mengintepretasikan berdasarkan fakta yang terkumpul (Endriss, 2005). Prolog merupakan bahasa pemograman yang menggunakan simbol dan komputasi non numerik, solusi didapat berdasarkan refensi dari

objek fakta dan relasi antara objek fakta dengan tujuan penyelesaikan masalah (Bratko, 2001). Dalam referensi lain, prolog juga disebut sebagai bahasa pemograman relasional, terdapat pemaparan hubungan antara fakta – fakta yang terkumpul dengan masalah yang akan diselesaikan (Huntbach *et al.*, 1996; Bramer, 2005).

Dalam metode penyelesaian masalah, prolog menggunakan *section facts* untuk mengumpulkan sejumlah fakta dan aturan berupa *database. Database* berupa fakta – fakta yang diperoleh untuk dijadikan refensi pemberi jawaban terhadapa masalah yang ada. Dalam prolog secara prosedural dijabarkan bagaimana komputer dapat menyelesaikan masalah dan masalah apa yang dapat diselesaikan berdasarkan referensi database yang sudah dibentuk (Smith, 2004). *Database* yang sudah terbentuk menjadi referensi dari jawaban atas permasalahan yang diajukan. *Section facts* memberikan jawaban berdasarkan *database* yang telah dibuat sebelumnya. Jawaban dan waktu penyelesaian masalah tergantung dari kompleksitas data yang telah dikumpulkan *database*. Semakin banyak data yang menjadi referensi jawaban, maka jawaban menjadi akurat dan diperoleh dengan sangat cepat. *Database* disusun dari sejumlah fakta, data dan pengalaman dari sumber data. Penggunaan metode *filtering* dimungkinkan untuk mengarahkan jawab kepada data yang spesifik.

Penyelesaian masalah dalam artificial intelligence menggunakan tindakan berurutan untuk mendeskripsikan sebuah fakta dan relasi terhadap masalah dalam keadaan *state* (Russell *et al.*, 1995). Untuk mencapai tujuan, *artificial intelligence* menggunakan agen intelegensi untuk melewati keadaan *sequence state* untuk memaksimalkan pengukuran performansi.

METODE

Tahap Penyelesaian Masalah

Tahap untuk menyelesaikan masalah dimulai dari tahap pencarian oleh agen penyelesaian masalah (*Problem Solving Agent*) yang disesuaikan dengan perumusan tujuan (*goal formulation*). Penyelesaian masalah dilakukan berdasarkan referensi data yang sudah dikumpulkan sebelumnya. *Problem Solving Agent* melakukan definisi elemen dari *problem* (masalah) dan *solution* (jawaban). Masalah didefinisikan sebagai sesuatu hal yang akan dipecahkan sedangkan solusi terkait dengan perumusan tujuan (*goal formulation* yang disebut sebagai *problem formulation. Problem formulation* adalah proses untuk menentukan tindakan dan keadaan apa yang harus dipertimbangkan berdasarkan perumusan tujuan. *Problem solving sequence* terdiri perumusan masalah (*formulate*), pencarian *(search*), eksekusi (*execute*) (Russell *et al.*, 1995).

```
function SIMPLE-PROBLEM-SOLVING-AGENT(p) returns an action
  inputs: p, a percept
  static: s, an action sequence, initially empty
          state, some description of the current world state
          g, a goal, initially null
          problem, a problem formulation

  state ← UPDATE-STATE(state, p)
  if s is empty then
      g — FORMULATE-GOAL(state)
      problem ← FORMULATE-PROBLEM(state, g)
      s — SEARCH(problem)
  action — RECOMMENDATION(s, state)
  s ← REMAINDER(s, state)
  return action
```

Gambar 1. *Problem Solving Agent*
(Sumber : Russell *et al.*, 1995)

Problem solving agent memilih dan mencari beberapa pilihan referensi dari kemungkinan tahapan penyelesaian masalah (*search*). Algoritma pencarian (*search*) memilih masalah sebagai masukan (*input*) untuk dimasukkan ke dalam urutan tindakan menuju penyelesaian masalah (*solution*). Proses pencarian alternatif pilihan jalur untuk menyelesaikan masalah berdasarkan resiko dan biaya jalur yang dikeluarkan (*path cost*).

```
datatype PROBLEM
  components: INITIAL-STATE, OPERATORS, GOAL-TEST, PATH-COST-FUNCTION
```

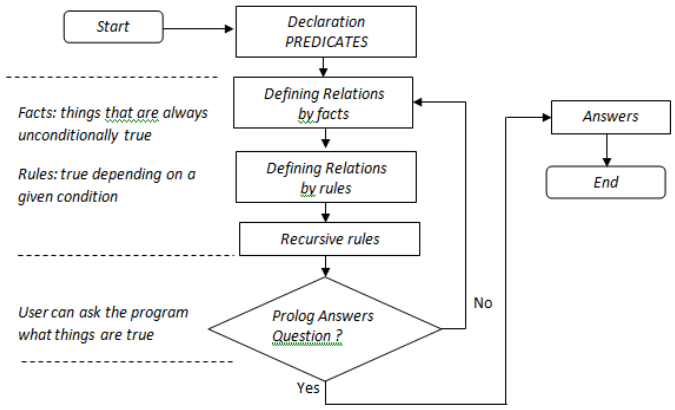
Gambar 2. Proses Penentuan Jalur
Berdasarkan *Path Cost*
(Sumber : Russell *et al.*, 1995)

Bahasa Pemograman Prolog

Bahasa prolog adalah bahasa pemograman yang bersifat logika. Bahasa prolog menggunakan teknik pencarian heuristik (*heuristic*). Penyelesaian masalah dalam prolog menggunakan pohon logika. Perbedaan dengan bahasa pemograman yang lain adalah bahasa pemograman selain prolog menggunakan algoritma konvensional, dimana didasarkan pada suatu algoritma yang disusun dengan jelas, rinci, serta langkah sampai pada hasil yang sudah ditentukan sebelumnya. Prolog mempunyai karateristik yang bersifat *declarative language* sehingga tidak memerlukan prosedur untuk menyelesaikan masalah. Prolog menampilkan fakta – fakta dalam bentuk data-data object yang akan diolah menjadi relasi antar object sehingga akan menjadi sebuah aturan.

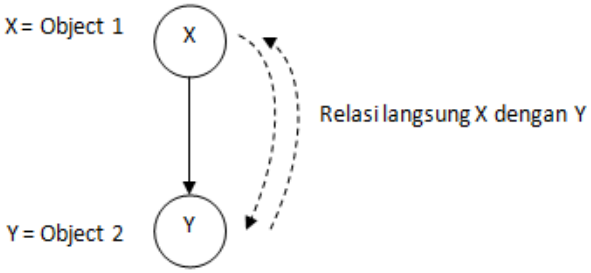
Dalam prolog terdapat tiga komponen penting untuk membentuk bahasa pemograman, PREDICATES, CLAUSES, dan GOAL. PREDICATES merupakan struktur *object* yang berisi tentang relasi antar *object*, dimana relasi tersebut digunakan untuk mendeklarasikan predikat. CLAUSES merupakan aturan dan fakta-fakta atau object yang akan dijadikan sebuah *database* sebagai referensi jawaban penyelesaian masalah. GOAL adalah tujuan atau jawaban yang akan dicari. Proses penyelesaian masalah pada prolog menggunakan metode *Formal Reasoning*, dengan membuktikan cocok tidaknya tujuan dengan data-data yang telah ada dan relasinya. Dari pencocokkan tersebut maka dapat ada didapatkan jawaban sesuai dengan yang diinginkan. Jika prolog menginginkan sebuah jawaban yang lebih spesifik maka prolog menggunakan fungsi *filter* pada struktur *section facts*. Prolog dapat menyelesaikan masalah dengan cara selayaknya manusia menyelesaikan permasalahan karena berdasarkan logika.

Sebelum penerapan *section facts* dalam prolog maka diperlukan aturan untuk menjelaskan sebuah relasi terhadap PREDICATES, CLAUSES dan GOAL. CLAUSES dalam prolog berisi *facts* dan *rules*. *Facts* merupakan sekumpulan fakta yang menjadi data referensi yang memuatkan sebuah relasi variabel dalam *facts*. Untuk menghasilkan sebuah jawaban maka dibutuhkan *rules* sebagai bentuk syarat dari proses pengolahan data.



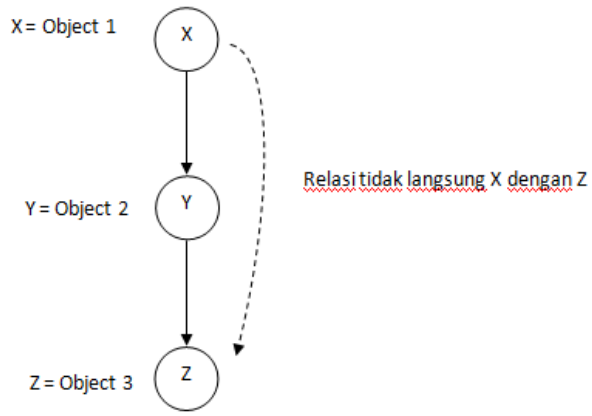
Gambar 3. Diagram Alir Prolog
(Sumber : Htay, 2016; Bratko, 2001)

Proses deklarasi relasi *facts,* adalah untuk melihat apakah kumpulan *facts* yang menjadi data referensi memiliki relasi antar objek. Proses ini dilakukan sebagai tahap awal untuk melihat seberapa dekat hubungan antara dua atau lebih objek yang dikumpulkan.



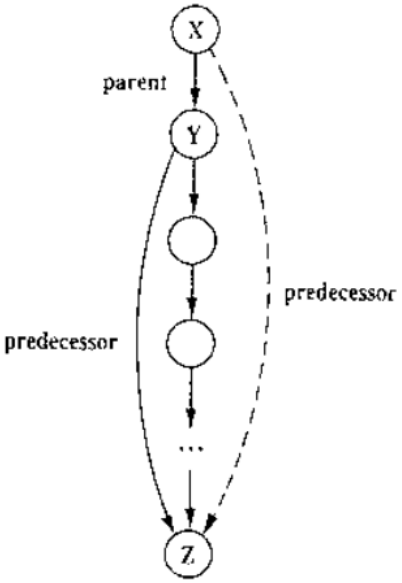
Gambar 4. Relasi antar objek dalam prolog hubungan langsung X-Y

Dalam prolog ditulis dengan ditulis dengan "*relation(X,Y)*", dimana X adalah objek 1 dan Y adalah objek 2. Relasi antara keduanya (X - Y) merupakan hubungan langsung yang menjelaskan tingkat kedekatan hubungan X dan Y. Dalam beberapa kasus ditemukan bahwa terdapat hubungan tidak langsung antara kedua objek (X-Y-Z). Objek Y ada jika adanya hubungan antara X dan Z.



Gambar 5. Relasi antar objek dalam prolog dengan hubungan antara X-Z

Menurut Bratko (2001) bahwa hubungan relasi antara objek dapat berupa hubungan langsung dan hubungan tidak langsung yang ditunjukkan pada *rules* seperti pada Gambar 6 berikut :



Gambar 6. *Recursive Formulation* Hubungan Objek (Sumber : Bratko, 2001)

Setelah proses untuk deklarasi rules maka prolog akan mencari jawaban yang sesuai dengan pertanyaan yang diajukan. Proses ini dilakukan pada tahapan *declarative meaning* yang menentukan bagaimana output diperoleh dan bagaimana relasi antar objek dievaluasi dalam prolog.

Penyelesaian Masalah Prolog dengan Metode Section Fact

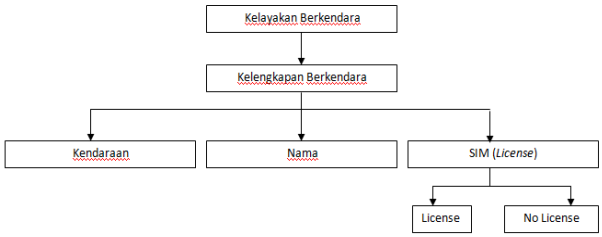
Section facts adalah kumpulan fakta dan aturan yang dapat digunakan kembali pada saat proses pencocokan jawaban yang sedang berlangsung. Proses section facts yang dimaksud adalah pemberian parameter variabel baru dimungkinkan. Section facts pada prolog berupa *database* yang dapat diisi, diubah dan dihapus seiring dengan dijalankannya program dengan fakta baru tersebut dituliskan pada section goal. Pada penelitian ini dilakukan dengan melakukan deklarasi *facts* berdasarkan pada parameter object. Proses ini dilakukan dengan metode filtering dimana prolog mencari jawaban dengan berdasarkan parameter mana yang menjadi acuan. Acuan yang dipakai merupakan salah satu parameter kategori dari objek yang dideklatasikan, semisal Kategori 1 terdiri dari objek A, objek B, objek C.

```
CLAUSES
        Category ("Object A", "Object B", "Object C")
```

Gambar 7. Section Facts Prolog

Berikut proses penerapan *section facts* pada penelitian ini dengan menggunakan bahasa

pemograman prolog pada kasus penentuan kelayakan berkendara.



Gambar 8. Kerangka konseptual kategori

Untuk menentukan kelayakan berkendara, maka perlu dibuat beberapa objek seperti Jenis Kendaraan, Identitas (nama) dan SIM sebagai bagian dari kategori kelengkapan berkendara. SIM dibagi menjadi 2 data yaitu "*License"* dan "*No License"*. Proses list bahasa pemograman sebagai berikut :

```
DOMAINS
        object A , object B = symbol
        object C= string
        list = object C*

PREDICATES
        nondeterm category(object A, object B, object C)
        nondeterm rules 1(object A, object B)
        nondeterm rules 2(object A, object B)

CLAUSES
        category("Data object A1", " Data object B1", " Data object C1").
        category("Data object A2", " Data object B2", " Data object C2").
        category("Data object A3", " Data object B3", " Data object C3").
                                .
                                .
                                .
        category("Data object An", " Data object Bn", " Data object Cn").
```

Gambar 9. Deklarasi *object* pada *section facts*

Setelah melakukan deklarasi *object* maka dilakukan identifikasi berdasarkan *rules.* Berikut *rules* yang dibuat untuk *section facts*.

```
rules 1 (X,Y):-
        category(X,Y, "license").
rules 2 (X,Y):-
        category(X,Y, "license").
```

Gambar 10. Deklarasi aturan (rules)

Proses deklarasi *fact* dan *rules* tersebut kemudian digunakan untuk melakukan proses filtering terhadap kategori yang ditentukan. Hasil yang didapat merupakan list data referensi dengan kategpri object "*license*".

HASIL DAN PEMBAHASAN

Dalam penelitian ini *section fact* dilakukan untuk mencocokkan jawaban berdasarkan fact dan aturan yang sudah dideklarasikan sebelumnya. Pada *section* DOMAIN diperkenalkan *object* "kendaraan" dan "nama" sebagai data yang memiliki file type Symbol, "sim" memiliki file type STRING. Command list pada DOMAIN digunakan untuk menampilkan data referensi berdasarkan *facts* yang sudah dikumpulkan.

```
DOMAINS
        kendaraan, nama = symbol
        sim = string
        list = sim*
```

Gambar 11. Section DOMAIN

Pada PREDICATES menampilkan struktur / bentuk dari data yang akan ditampilkan di CLAUSES. Predicates

menampilkan struktur hubungan / relasi antar objek pada CLAUSES

```
PREDICATES
        nondeterm kelengkapan_berkendara(kendaraan, nama, sim)
        nondeterm layak_berkendara(kendaraan, nama)
        nondeterm tidak_layak_berkendara(kendaraan, nama)
```

Gambar 12. Section PREDICATES

Penambahan command nondeterm pada PREDICATES digunakan karena memungkinkan pada bagian ini untuk dapat menampilkan jawaban lebih dari satu jawaban yang akan muncul.

Pada *section* CLAUSES akan dideklarasikan sekumpulkan facts yang nanti akan dijadikan sebagai data referensi.

```
CLAUSES
        kelengkapan_berkendara("Avanza", "Joko", "A").
        kelengkapan_berkendara("Xenia", "Joni", "A").
        kelengkapan_berkendara("Jeep", "Jodi", "A").
        kelengkapan_berkendara("Supra Fit", "Joki", "C").
        kelengkapan_berkendara("Becak", "Anton", "No License").
        kelengkapan_berkendara("Pesawat", "Adi", "Pilot License").
        kelengkapan_berkendara("Mobil box", "Aji", "B1").
        kelengkapan_berkendara("Fortuner", "Bardi", "A").
        kelengkapan_berkendara("Vixion", "Badu", "C").
        kelengkapan_berkendara("Tiger", "Banjo", "C").
```

Gambar 13. Deklarasi *facts* pada *section* CLAUSES

Sedangkan untuk menentukan jawaban maka diberikan sebuah *rules*

```
layak_berkendara(X,Y):-
        kelengkapan_berkendara(X,Y,"A");
        kelengkapan_berkendara(X,Y,"C").

tidak_layak_berkendara(X,Y):-
        kelengkapan_berkendara(X,Y,"No License");
        kelengkapan_berkendara(X,Y,"Pilot License").
```

Gambar 14. Deklarasi rules

Rules merupakan syarat atau aturan yang menyeleksi facts berdasarkan relasi untuk menjawab dari pertanyaan yang diajukan dalam bahasa pemograman biasanya ditulis dengan “If , Then” yang disimbolkan dengan “:-“ pada prolog . Penelitian ini mempunyai tujuan untuk menerapkan section facts dengan menampilkan semua data yang sesuai dengan kategori yang di persyaratkan yaitu layak berkendara.

```
[Inactive C:\Users\Habibie\AppData\Local\Temp\goal$000.exe]
Kendaraan=Avanza, Nama=Joko
Kendaraan=Xenia, Nama=Joni
Kendaraan=Jeep, Nama=Jodi
Kendaraan=Fortuner, Nama=Bardi
4 Solutions
```

Gambar 15. Hasil Output Prolog

KESIMPULAN

Prolog merupakan bahasa pemograman deklaratif dan berdasarkan relasi antar objek. Dalam deklarasi PREDICATES ditentukan sebuah struktur dari data. Data yang dimaksud adalah referensi *facts* yang digunakan sebagai data CLAUSES. *Section fact* menetapkan sebuah kategori yang akan ditampilkan berupa list data. Section facts berisi aturan dan facts yang digunakan untuk mencocokkan jawabn dengan relasi objek pada CLAUSES, sedangkan metode filtering merupakan prolog mencari jawaban dengan berdasarkan parameter mana yang menjadi acuan.

DAFTAR PUSTAKA

Bramer, M. 2005. *Logic Programming with Prolog*. Portsmouth: Springer.

Bratko, I. 2001. *Prolog - programming for artificial intelligence*. 3rd ed. Addison - Wesley.

Endriss, U. 2005. *Introduction to Prolog Programming*. Amsterdam: Institute for Logic, Language and Computation Lecture.

Htay, M.M. 2016. *Prolog Programming*.

Huntbach, M., Science, C., Mary, Q. & College, W. 1996. Artificial Intelligence 1 Notes on Prolog. (5): 1–34.

Kurniawan, C. 2016. Implementasi Artificial Intelligence Dalam Penyelesaian Masalah Dengan Metode Unification Dan Back Tracking Visual Prolog. *SINTEKS*, 1–15.

Russell, S.J., Norvig, P., Canny, J.F., Malik, J.M., Edwards, D.D., Jonathan, S.J.S. & Norvig, P. 1995. *Artificial Intelligence : A Modern Approach*. New Jersey: Prentice Hall.

Smith, T. 2004. *Artificial Intelligence Programming in Prolog*.