Introductie tot Aleph

Intelligente Systemen 2012-2013

Met dank aan Mihai Polak

In het vijfde practicum maken we gebruik van Aleph, een Inductive Logic Programming (ILP) library voor Prolog. Dit document is bedoeld als korte introductie tot Aleph. Uitgebreide documentatie vindt je op de homepage: `http://www.cs.ox.ac.uk/activities/machlearn/Aleph/aleph.html`.

1 YAP

Omdat er wat incompatibiliteit bestaat tussen Aleph en SWI-Prolog, maken we voor dit practicum gebruik van een andere Prolog interpreter: YAP (Yet Another Prolog, `http://www.dcc.fc.up.pt/~vsc/Yap/`). De meeste commando's komen overeen met SWI-Prolog, maar YAP draait in de commandline in plaats van in een eigen window. Om installatie en gebruik te vergemakkelijken is er een zip-file op de practicumpagina gepubliceerd met daarin alle benodigdheden en wat instructies.

2 Invoer

We zullen laten zien hoe je Aleph gebruikt aan de hand van een voorbeeld uit het boek. Je kunt dit voorbeeld ook vinden in het archief met de practicumbestanden. We nemen een familieboom als achtergrondkennis en willen het predicaat `grandfather(X,Y)` afleiden, wat aangeeft dat `X` grootvader is van `Y`.

De input bestaat uit drie bestanden met dezelfde naam, maar met verschillende extensies: `<naam>.b`, `<naam>.f` en `<naam>.n`. Het .n bestand zullen we in dit practicum leeg laten[1].

[1] Het .n bestand is bedoeld voor negatieve voorbeelden.

2.1 .b

Het .b bestand vertelt Aleph wat er geleerd moet worden, hoe het leerproces moet verlopen en welke kennis al bekend is.

2.1.1 Determinations

Determinations geven aan voor welke predicaten we regels willen leren en welke predicaten daarvoor gebruikt mogen worden. Voor elk te leren predicaat moet dit apart gedaan worden. Merk op dat dit niet betekent dat al die predicaten ook daadwerkelijk in de geleerde regels voor moeten komen, het zijn slechts mogelijk relevante predicaten. Voor `grandfather/2` kunnen we het volgende gebruiken:

```
:- determination(grandfather/2, person/1).
:- determination(grandfather/2, father/2).
:- determination(grandfather/2, mother/2).
:- determination(grandfather/2, male/1).
:- determination(grandfather/2, female/1).
```

2.1.2 Modes

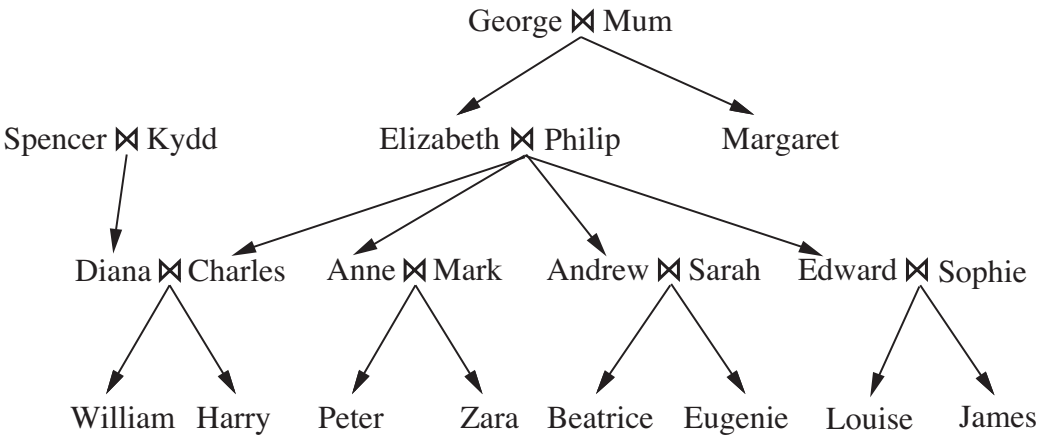
Mode declarations geven aan hoe een predicaat aangeroepen moet worden. Ze definiëren voor alle argumenten of ze input of output zijn en van welk type ze zijn. Er zijn twee varianten van: `modeh` voor predicaten die in de head van een regel moeten komen te staan (i.e. de consequent van de implicatie) en `modeb` voor predicaten in de body (antecedent van de implicatie). In ons voorbeeld hebben we de volgende mode declarations:

```
:- modeh(*,grandfather(+person, -person)).

:- modeb(*,father(+person, -person)).
:- modeb(*,mother(+person, -person)).
:- modeb(*,male(+person)).
:- modeb(*,female(+person)).
```

Het eerste argument heeft met non-determinantie te maken en mag je altijd op `*` laten staan. De types zijn unaire predicaten die in de background knowledge moeten staan. Merk op dat `?argument` anders dan in de practicumopgaven hier geen betekenis heeft. Als je zoiets toch wilt definiëren, kun je een predicaat meerdere mode declarations geven.

2.1.3 Background knowledge



Verder wordt in het .b bestand de achtergrondkennis gegeven. Hierin staan alle regels die Aleph niet zelf hoeft te leren, bijvoorbeeld de definitie voor `mother/2`. Hier horen ook de types bij die in de mode declarations gebruikt worden. In ons voorbeeld:

```
% types
person(george).
person(mum). %etc...

% background knowledge
father(george, elizabeth).
father(george, margaret). %etc...

mother(mum, elizabeth).
mother(mum, margaret). %etc...

male(george).
male(philip). %etc...

female(elizabeth).
female(diana). %etc...
```

Merk op dat Aleph zelf geen onderscheid maakt tussen types en ander soort achtergrondkennis. Merk verder op dat het ook gewoon toegestaan is om complexere regels te gebruiken, `person` hadden we hier dus ook kunnen definiëren in termen van `male` en `female`.

2.2 .f

Om te induceren / leren hoe de definitie van het doelpredicaat eruit kan zien, heeft Aleph voorbeelden nodig. Die worden gespecificeerd in het .f bestand (de positieve voorbeelden) en in het .n bestand (de negatieve voorbeelden). In dit practicum gebruiken we alleen positieve voorbeelden. Het .f bestand bij de family tree kan er als volgt uitzien.

```
grandfather(george, andrew).
grandfather(philip, zara).
grandfather(spencer, harry). %etc...
```

Het is belangrijk dat de voorbeelden representatief zijn. Als alle grootvaders die Aleph heeft gezien alleen kleinkinderen hebben via hun dochters, zou het af kunnen leiden dat het hebben van een dochter vereist is voor het grootvaderschap. Denk er dus over na bij het maken van de opdrachten dat je Aleph niet onbedoeld misleidt.

3 Executie

Om Aleph uit te voeren volg je de volgende stappen:

1. Open `OpenYAP.bat` in de lib folder (of start YAP zelf op uit de commandline, maar zorg wel dat je dit vanuit de lib folder doet). We zijn nu in een Prolog-environment, net als SWI-Prolog.
2. Laad Aleph: `[aleph].`
3. Gebruik: `read_all(grandfather).` om de grandfather bestanden in te lezen.
4. Gebruik: `induce.` om de gewenste regels af te leiden.

4 Uitvoer

Zodra je `induce.` hebt gedaan, krijg je precies te zien wat Aleph allemaal gedaan heeft. Bij grandfather ziet dit er als volgt uit:

1. Aleph kiest een van de voorbeelden uit het .f bestand.

```
[select example] [1]
[sat] [1]
[grandfather(george,andrew)]
```

2. Voor dat voorbeeld bouwt het een hele specifieke regel.

```
[bottom clause]
grandfather(A,B) :-
        father(A,C), father(A,D), male(A),
        mother(C,B), female(C).
```

3. Aleph probeert de regel algemener te maken om zo meer van de voorbeelden af te vangen.

```
[reduce]
...
grandfather(A,B) :-
        father(A,C).
...
grandfather(A,B) :-
        male(A).
```

4. Aleph vindt een regel die lijkt te kloppen. `pos-cover` geeft aan hoeveel van de voorbeelden eruit afleidbaar zijn.

```
[found clause]
grandfather(A,B) :-
        father(A,C), mother(C,B).
[pos-cover = 4 ...]
```

5. Aleph kiest de beste regel, slaat deze op en gaat terug naar 1 totdat alle voorbeelden uit de gevonden regels volgen.

```
[best clause]
grandfather(A,B) :-
        father(A,C), mother(C,B).
...
[positive examples left] [2]
```

6. Wanneer er geen voorbeelden meer over zijn, geeft Aleph een overzicht van de gevonden regels.

```
[theory]

[Rule 1] ...
grandfather(A,B) :-
        father(A,C), mother(C,B).

[Rule 2] ...
grandfather(A,B) :-
        father(A,C), father(C,B).
```