

Документация към проект  
Интерпретатор на *Prolog*  
ФП, зимен семестър 2022/2023

Калоян Цветков  
4MI0800017

ФМИ, СУ  
1.0.0

# Съдържание

<b>1</b>	<b>Кратко описание</b>	<b>2</b>
<b>2</b>	<b>Структура</b>	<b>2</b>
<b>3</b>	<b>Основни концепции и функции</b>	<b>2</b>
3.1	Примерен входен файл . . . . .	2
3.2	Унификация на термове . . . . .	3
3.3	Резолуция . . . . .	4
3.3.1	Примерно дърво . . . . .	5
<b>4</b>	<b>Работа с приложението</b>	<b>6</b>
4.1	Начин на работа . . . . .	6
4.2	Примерна работа . . . . .	6

## 1 Кратко описание

Интерпретаторът е написан на езика Haskell и компилиран чрез GHC, версия 8.10.7.

Може да определя истинността на факти, заредени от програма на езика *Prolog*, използвайки директно записаните в него константни факти или използвайки изводи от правила.

При подаване на факт, съдържащ променлива (заявка), интерпретаторът определя всички стойности на променливата, за които фактът е верен (ако има такива според кода) и при поискване ги принтира. Също така може да разрешава равенства на термове, отново посочвайки за кои стойности на променливите равенството е изпълнено.

Интерпретаторът е изпълним файл, който се стартира в конзола.

## 2 Структура

Сорс-кодът е разделен в 8 модула:

- **Prolog**  
Съдържа всички останали модули;
- **Datatypes**  
Съдържа всички потребителски типове, представляващи елементите на една програма на *Prolog*;
- **Checkers**  
Предикати, определящи дали низ може да се конвертира до синтактичен елемент от *Prolog* и други;
- **Conversions**  
Функции за конвертиране между синтактични елементи на *Prolog*;
- **Identities**  
Функции за проверка на идентичност на потребителски типове;
- **Tools**  
Общи функции за обработка на данни;
- **Unification**  
Реализация на алгоритъма за унификация на термове
- **Resolutions**  
Основен метод; съдържа функции за проверка на истинност на факти.

Файлът `main.hs` съдържа само функции за работа с външния свят (вход и изход).

## 3 Основни концепции и функции

### 3.1 Примерен входен файл

Входен файл: *prolog/starwars.pl*:

---

```
%father/2
father(ruwee, padme).
father(anakin, luke).
father(anakin, leia).
father(han, ben).
```

---

1  
2  
3  
4  
5  
6

```

%mother/2
mother(jobal, padme).
mother(shmi, anakin).
mother(padme, luke).
mother(padme, leia).
mother(leia, ben).

%alias/2
alias(darthvader, anakin).
alias(kyloren, ben).
alias(X,Y) :- alias(Y,X).

%parent/2
parent(X,Y) :- father(X, Y).
parent(X,Y) :- mother(X, Y).

%parent/2
childof(X,Y) :- parent(Y,X).

```

### 3.2 Унификация на термове

За унификация на два терма се грижи функцията

---

```
toBeUnified :: (Term, Term) -> QueryResult
```

---

Функцията имплементира следния алгоритъм за унификация:

---

```

Input: Terms A, B
Initialize the result list to be empty;
Initialize the stack with pair (A,B);
Initialize the fail flag to false;
While stack is not empty and not fail do:
  pop (X,Y) from the stack
  Case
  X is a variable:
    Substitute X with Y in the stack
    add X=Y to the list
  Y is a variable:
    Substitute Y with X in the stack
    add Y=X int the list
  X and Y are identical constants
    continue
  X is atom f(X_1,...,X_n) and Y is atom f(Y_1,...,Y_n) for some identifier f and
    n>0:
    for i=1..n: push (X_i,Y_i) in the stack
  otherwise:
    set fail to true
  end Case
end While
Output:
If fail, then empty QueueResult
else QueryResult, corresponding to the result list;

```

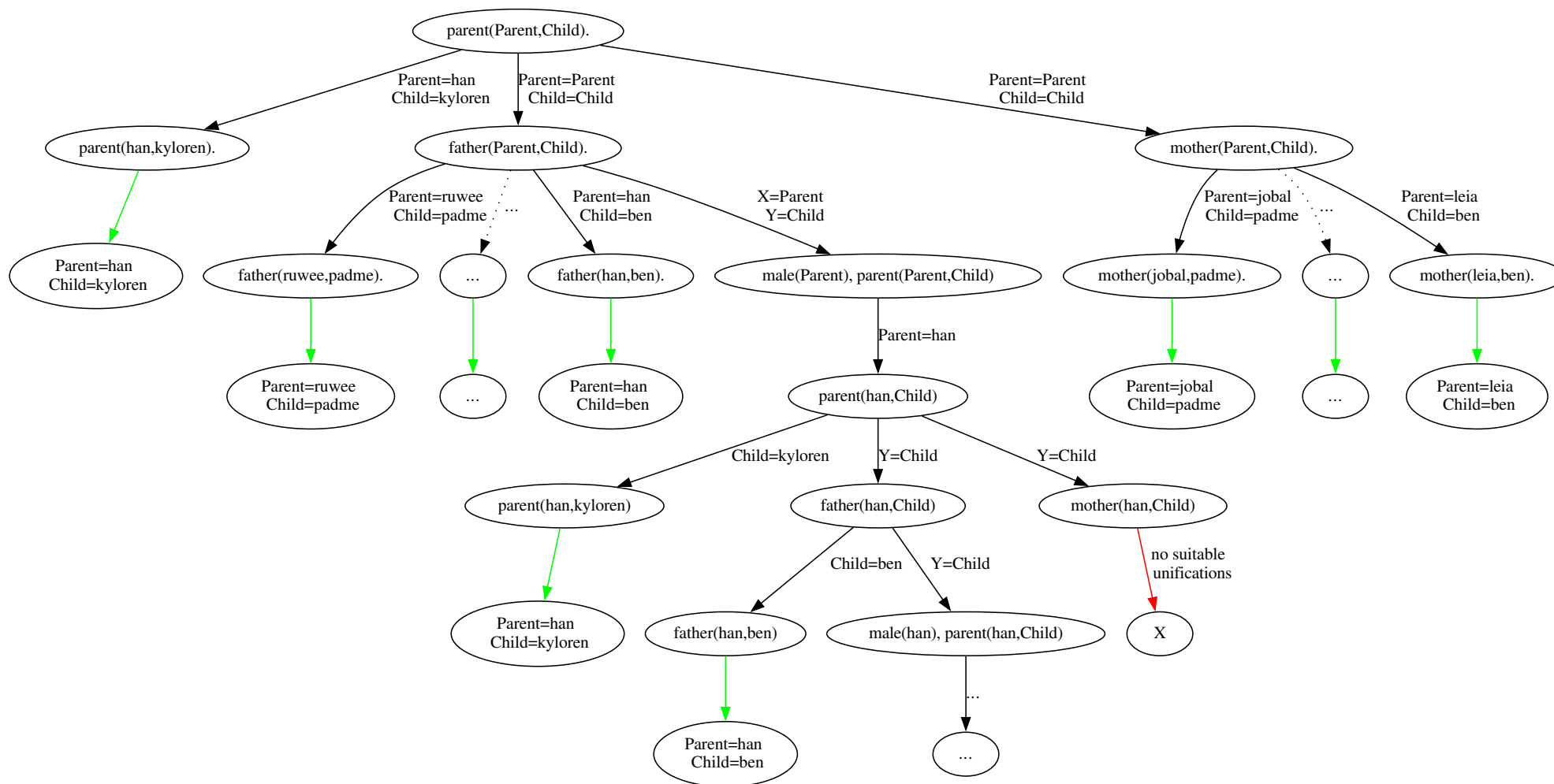
---

### 3.3 Резолюция

Резолюцията на термове се извършва по алгоритъма на *SLD резолюцията*. Строи се кореново дърво, чиито върхове съдържат (на практика не е необходимо) конюнкция на термове. Листата на дървото съдържат необходими преобразувания до достигане на верен факт от заявката. Дървото е потенциално безкрайно, но благодарение на механизма за нормално оценяване, самото конструиране на дървото и извличане на резултати от него стават паралелно.

#### 3.3.1 Примерно дърво

Заявката е *"parent(Parent,Child)"*:



## 4 Работа с приложението

### 4.1 Начин на работа

След стартиране, приложението изисква да се въведе от клавиатурата път до код на *Prolog*. При въведен синтактично неправилен низ или път, който не води към файл, приложението пита повторно дали потребителят желае да въведе нов път. Ако пътят е валиден, всички факти в него биват проверени за синтактична вярност и се изписва съобщение с резултата (неверните редове биват изведени и в последствие не биват разглеждани).

След това започва *Read-Eval-Print-Loop*, очакващ потребителят да въведе равенство, факт или заявка, и съответно интерпретаторът изписва резултатите.

Работата с текущия файл приключва след подаване на команда *quit*.

### 4.2 Примерна работа

Стартиране на интерпретатора (в конзолен режим):

---

```
Which file to consult from the directory "prolog/"?
> starwar.pl
No such file...
Consult another file? ( y | [n] )
> y
Which file to consult from the directory "prolog/"?
> starwars.pl
true.
> father(anakin,luke).
true.
> father(anakin,luke)
You are allowed to input only facts, queries and equalities!
> father(luke,anakin).
false.
> father(Parent,Child).
Parent = ruwee.
Child = padme.

Parent = anakin.
Child = luke.

Parent = anakin.
Child = leia.

Parent = han.
Child = ben.

Parent = han.
Child = kyloren.
.
> a(X)=a(arg).
X = arg.
> a(Y,r)=b(X,Z).
false.
> a(Y,r)=a(X,Z).
Y = X.
```

---

```
Z = r.  
> pred(Z,Z)=pred(X,t).  
X = t.  
Z = t.  
> pred(Z,Z)=pred(X,Y).  
Y = X.  
Z = Y.  
> quit  
Consult another file? ( y | [n] )  
> no  
Closing...
```

---