DD1351 LAB2 2018
DIEGO LEON 961206
VIKTOR MEYER 971203

Rapport Beviskontroll

Övergripande implementation

Enligt labblydelsen används ett indata format som består av en lista med premisser, ett bevismål, samt ett utfört bevis med naturlig deduktion. Prologprogrammet som vi ska bygga och uppgiften för den här labben är alltså att undersöka om ett bevis är korrekt eller inte.

Ett bevis kan sägas vara korrekt om alla rader i det angivna beviset stämmer överrens med de anginva reglerna för naturlig deduktion, reglerna som kommer att användas relaterar direkt till kursboken. Det får inte förekomma några odefinierade eller inkorrekta operationer i ett korrekt bevis. Utföver detta krav på korrekta operationer måste ett korrekt bevis sluta i det angivna målet från indata. Det är även intuitivt att ett korrekt bevis heller inte får avslutas på ett antagande.

De mer tekniska delarna av implementationen sker i Prolog med hjälp av predikatlogik. Beviskontrollsprogrammet startar som tidigare nämnt med indata i form av en lista med premisser, ett bevismål samt ett bevis uttryckt som en lista där varje element motsvarar en rad i beviset.

Strategin för den implementation som används i syfte att lösa denna labb är att gå igenom varje element i indatabeviset, det vill säga hantera rad för rad. I det fall att en rad visat sig vara korrekt kommer programmet att lägga till den giltiga raden i en ny lista av ‘bevisade/färdiga’ rader, lämpligt namngiven ‘proved’. De nästkommande raderna i indatabeviset har möjlighet att använda sig av de tidigare bevisade raderna från ‘proved’ men självklart inte de andra ännu ej bevisade raderna i indatabeviset då detta hade varit inkorrekt.

Naturligt uppstår frågan om vad en korrekt rad egentligen är, och likt tidigare specificerat används reglerna för naturlig deduktion från boken. Varje regel implementeras i form av ett predikat som matchar mot en specifik regel och dess struktur. Predikaten har i sin tur definierats enligt vad som faktiskt är en giltig operation per regel.

För att snabbt sammanfatta är strategin att givet indatabeviset gå igenom rad för rad. Samt att för varje rad verfiera raden med hjälp av väldefinierade predikat och att efter alla rader hanterats även verfiera att målet faktiskt uppnåts.

```
Indata:
Premisser - Lista av alla premisser
Bevismål - Mål för beviset
Bevis - Lista av alla rader i beviset

Utdata:
Yes / No, där Yes motsvarar korrekt bevis och Nej motsvarar icke
korrekt bevis
```

DD1351 LAB2 2018
DIEGO LEON 961206
VIKTOR MEYER 971203

Hantering av boxar

Antaganden i beviset vi får in från textfilen finns i form av listor. Om vårt predikat valid_proof stöter på ett antagande går den rekursivt genom listan och slår slutligen ihop den med lisan Proved. Därefter går man vidare till nästa rad i beviset.

För att få tag på specifika rader i beviset används det inbyggda predikatet member som givet ett element och en lista söker efter ett elementet i en lista och returnerar true om elementet finns i listan. Om det sökta elementet finns i listan men befinner sig inom en annan lista, så returnerar member false då tex: [1, q, premise] inte är lika med [[1, q, premise]]. Detta sker då den sökta raden befinner sig i en “assumption box”. Då predikaten för reglerna:
PBC, orel, negint, impint, bör ha tillgång till rader från assumption boxar används följande predikat som effektivt ger tillgång till en box:

```
getBox(X, [L|_], L):- member([X, _, _], L).
getBox(X, [_|T], L):- getBox(X, T, L).
```

Predikatet getBox har som uppgift att hitta en assumption box i Proved som innehåller den sökta raden. Först söker predikatet efter en box(inte subboxar) och tar hjälp predikatet member och det givna radnumret för att kontrollera om raden finns i boxen. Om member returnerar false söker man efter nästa assumption box . I fall predikatet returnerar true så returnerar getBox assumption boxen.

Härledning av predikat

Premiss

```
valid_proof(Prems, Goal, [[RowNumb, Formula, premise] | T], Proved, InBox) :-
member(Formula, Prems), valid_proof(Prems, Goal, T, [[RowNumb, Formula, premise] |
Proved], InBox).
```

Predikatet är sant om och endast om listan av indatapremisser innehåller formeln Formula.

Assumption

```
valid_proof(Prems, Goal, [[[RowNumb, Formula, assumption] | T2] | T1], Proved,
InBox) :- valid_proof(Prems, Goal, T2, [[RowNumb, Formula, assumption] | Proved],
1), valid_proof(Prems, Goal, T1, [[[RowNumb, Formula, assumption] | T2] | Proved],
InBox).
```

Predikatet är alltid sant då det alltid är tillåtet att göra ett antagande, ett specialfall är såklart att ett antagande inte får ske som slutsatts i ett bevis, men det specialfallet hanteras ej av detta predikat utan på ett annat sätt.

LEM

```
valid_proof(Prems, Goal, [[RowNumb, or(X, neg(X)), lem] | T], Proved, InBox) :-
valid_proof(Prems, Goal, T, [[RowNumb, or(X, neg(X)), lem] | Proved], InBox).
```

Predikatet är sant om och endast om den införda formeln matchar formen or(X, neg(X)).

Contradiction Eliminiation

```
valid_proof(Prems, Goal, [[RowNumb, Formula, contel(X)] | T], Proved, InBox) :-
member([X, cont,_], Proved), valid_proof(Prems, Goal, T, [[RowNumb, Formula,
contel(X)] | Proved], InBox).
```

Predikatet är sant om och endast om det på rad X uppstått en motsägelse.

And Introduction

```
valid_proof(Prems, Goal, [[RowNumb, and(Xval, Yval), andint(X, Y)] | T], Proved,
InBox) :- member([X, Xval,_], Proved), member([Y, Yval,_], Proved),
valid_proof(Prems, Goal, T, [[RowNumb, and(Xval, Yval), andint(X, Y)] | Proved],
InBox).
```

Predikatet är sant om och endast om den införda formeln matchar and(Xval, Yval) samt att Xval faktiskt finns på rad X och Yval faktiskt finns på rad Y, där X och Y är givna från andint(X,Y).

And Elimination 1

```
valid_proof(Prems, Goal, [[RowNumb, Formula, andel1(X)] | T], Proved, InBox) :-
member([X, and(Formula,_),_], Proved), valid_proof(Prems, Goal, T, [[RowNumb,
Formula, andel1(X)] | Proved], InBox).
```

Predikatet är sant om och endast om den införda formeln Formula matchar det första argumentet givet på rad X till en ‘and’ där X är given från andel1(X).

And Elimination 2

```
valid_proof(Prems, Goal, [[RowNumb, Formula, andel2(X)] | T], Proved, InBox) :-
member([X, and(_,Formula),_], Proved), valid_proof(Prems, Goal, T, [[RowNumb,
Formula, andel2(X)] | Proved], InBox).
```

Predikatet är sant om och endast om den införda formeln Formula matchar det andra argumentet givet på rad X till en ‘and’ där X är given från andel2(X).

Or Introduction 1

```
valid_proof(Prems, Goal, [[RowNumb, or(Xval, Yval), orint1(X)] | T], Proved, InBox)
:- member([X, Xval,_], Proved), valid_proof(Prems, Goal, T, [[RowNumb, or(Xval,
Yval), orint1(X)] | Proved], InBox).
```

Predikatet är sant om och endast om den införda formeln Formula har strukturen or(Xval) samt att det som är givet på rad X faktiskt är Xval där raden X är given från orint1(X).

Or Introduction 2

```
valid_proof(Prems, Goal, [[RowNumb, or(Xval, Yval), orint2(Y)] |
T], Proved, InBox) :- member([Y, Yval,_], Proved),
valid_proof(Prems, Goal, T, [[RowNumb, or(Xval, Yval), orint2(Y)]
| Proved], InBox).
```

Predikatet är sant om och endast om den införda formeln Formula har strukturen or(Yval) samt att det som är givet på rad Y faktiskt är Yval där raden Y är given från orint2(Y).

Implication Elimination

```
valid_proof(Prems, Goal, [[RowNumb, Formula, impel(X,Y)] | T], Proved, InBox) :-
member([X, Xval, _], Proved), member([Y, imp(Xval, Formula), _], Proved),
valid_proof(Prems, Goal, T, [[RowNumb, Formula, impel(X,Y)]|Proved], InBox).
```

Predikatet är sant om och endast om den införda formeln Formula matchar det andra argumentet givet på rad Y till en ‘imp’ samt att formeln Xval från rad X matchar det första argumentet till ‘imp’ på rad Y där X och Y är givna från impel(X, Y).

Implication Introduction

```
valid_proof(Prems, Goal, [[RowNumb, imp(Xval, Yval), impint(X,Y)] | T], Proved,
InBox) :- getBox(X, Proved, NewList), member([X, Xval, _], NewList), member([Y,
Yval, _], NewList), valid_proof(Prems, Goal, T, [[RowNumb, imp(Xval, Yval),
impint(X,Y)] | Proved], InBox).
```

Predikatet är sant om och endast om den införda formeln Formula har strukturen imp(Xval, Yval), där Xval överrenstämmer med det som faktiskt står på rad X, där YVal överrenstämmer med det som faktiskt står på rad Y, där raderna X och Y är givna från impint(X,Y).

Double Negation Elimination

```
valid_proof(Prems, Goal, [[RowNumb, Formula, negnegel(X)] | T], Proved, InBox) :-
member([X, neg(neg(Formula)),_], Proved), valid_proof(Prems, Goal, T, [[RowNumb,
Formula, negnegel(X)] | Proved], InBox).
```

Predikatet är sant om och endast om den införda formeln Formula matchar argumentet givet på rad X till en ‘neg(neg(Formula))’ där raden X är givnen från negnegel(X).

Double Negation Introduction

```
valid_proof(Prems, Goal, [[RowNumb, neg(neg(Formula)), negnegint(X)] | T], Proved,
InBox) :- member([X, Formula,_], Proved), valid_proof(Prems, Goal, T, [[RowNumb,
neg(neg(Formula)), negnegint(X)] | Proved], InBox).
```

Predikatet är sant om och endast om den införda formeln Formula har strukturen neg(neg(Formula)) samt att det som är givet på rad X faktiskt motsvarar Formula där raden X är givnen från negnegint(X).

Negation Elimination

```
valid_proof(Prems, Goal, [[RowNumb, cont, negel(X,Y)] | T], Proved, InBox) :-
member([X, Xval, _], Proved), member([Y, Yval, _], Proved), neg(Xval) = Yval,
valid_proof(Prems, Goal, T, [[RowNumb, cont, negel(X,Y)] | Proved], InBox).
```

Predikatet är sant om och endast om den införda formeln Formula är lika med cont samt att det som är givet på rad X är en direkt motsägelse mot det som är givet på rad Y där raderna X och Y är givna från negel(X,Y).

DD1351 LAB2 2018
DIEGO LEON 961206
VIKTOR MEYER 971203

Negation Introduction

```
valid_proof(Prems, Goal, [[RowNumb, neg(Xval), negint(X,Y)] | T], Proved, InBox) :-
getBox(X, Proved, NewList), member([X, Xval, assumption], NewList), member([Y,
cont, _], NewList), valid_proof(Prems, Goal, T, [[RowNumb, Formula, negint(X,Y)] |
Proved], InBox).
```

Predikatet är sant om och endast om den införda formeln har strukturen neg(Xval) samt att det på rad Y skett en motsägelse under förutsättningen Xval taget från rad X, där X och Y är givna från negint(X,Y).

Copy

```
valid_proof(Prems, Goal, [[RowNumb, Formula, copy(X)] | T], Proved, InBox) :-
member([X, Formula, _], Proved), valid_proof(Prems, Goal, T, [[RowNumb, Formula,
copy(X)] | Proved], InBox).
```

Predikatet är sant om och endast om den införda formeln Formula matchar formula givet på rad X samt att X är givnet från copy(X).

Or Elimination

```
valid_proof(Prems, Goal, [[RowNumb, Zval, orel(X, Y, Z, R, W)] | T], Proved, InBox)
:- getBox(Y, Proved, NewListY), getBox(R, Proved, NewListR), member([Y, Yval, _],
NewListY), member([R, Rval, _], NewListX), member([X, or(Yval, Rval), _], Proved),
member([Z, Zval, _], NewListY), member([W, Zval, _], NewListR), valid_proof(Prems,
Goal, T, [[RowNumb, Zval, orel(X, Y, Z, R,W)] | Proved], InBox).
```

Predikatet är sant om och endast om den införda formeln matchar formeln Zval på rad Z och rad W, getBox hittar en assumption box som innehåller rad X samt en annan box för rad R, member hittar en formel Yval på rad Y och Rval på rad R som bildar strukturen or(Yval, Rval) som matchar formeln på rad X samt att X, Y, Z, R, W är givna från orel(X, Y, Z, R, W).

Proof By Contradiction

```
valid_proof(Prems, Goal, [[RowNumb, Formula, pbc(X,Y)] | T], Proved, InBox) :-
getBox(X, Proved, NewList), member([X, neg(Formula), assumption], NewList),
member([Y, cont, _], NewList), valid_proof(Prems, Goal, T, [[RowNumb, Formula,
pbc(X,Y)]|Proved], InBox).
```

Predikatet är sant om och endast om getBox hittar en assumption box som innehåller rad X, den införda formeln Formula finns med på rad X som neg(Formula) och är av typen assumpiton, det finns en rad Y med formel “cont” samt att X och Y är givna från PBC(X, Y).

Modus Tollens

```
valid_proof(Prems, Goal, [[RowNumb, neg(Xval), mt(X,Y)] | T], Proved, InBox) :-
member([X, imp(Xval, Yval), _], Proved), member([Y, neg(Yval), _], Proved),
valid_proof(Prems, Goal, T, [[RowNumb, neg(Xval), mt(X,Y)]|Proved], InBox).
```

Predikatet är sant om och endast om den införda formeln matchar neg(Xval), att imp(Xval, Yval) faktiskt finns på rad X och att neg(Yval) finns på rad Y samt X och Y är givna från mt(X, Y).

Appendix

Korrekt bevis

```
[imp(and(p,q), r)].
imp(p, imp(q,r)).
[
  [1, imp(and(p, q),r),    premise],
  [
    [2, p,        assumption],
    [
      [3, q,        assumption],
      [4, and(p,q),        andint(2,3)],
      [5, r,        impel(4,1)]
    ],
    [6, imp(q,r),        impint(3,5)]
  ],
  [7, imp(p, imp(q,r)),          impint(2,6)]
].
```

Icke korrekt bevis

```
[imp(and(p,q), r)].
imp(p, imp(q,r)).
[
  [1, imp(and(p, q),r),    premise],
  [
    [2, p,        assumption],
    [
      [3, q,        assumption],
      [4, and(p,q),        andint(2,3)],
      [
        [5, r,        assumption]
      ]
    ],
    [6, imp(q,r),        impint(3,5)]
  ],
  [7, imp(p, imp(q,r)),          impint(2,6)]
].
```

Programkod

```
% import proof
verify(InputFileName) :- see(InputFileName),
read(Prems), read(Goal), read(Proof),
seen,
valid_proof(Prems, Goal, Proof, [], 0).

% no proof and nothing proved cant be a valid proof
valid_proof(Prems, Goal, [], [], InBox) :- !, fail.
```

DD1351 LAB2 2018
DIEGO LEON 961206
VIKTOR MEYER 971203

```
% premise
valid_proof(Prems, Goal, [[RowNumb, Formula, premise] | T], Proved, InBox) :-
member(Formula, Prems), valid_proof(Prems, Goal, T, [[RowNumb, Formula, premise] |
Proved], InBox).

% assumpiton
valid_proof(Prems, Goal, [[[RowNumb, Formula, assumption] | T2] | T1], Proved,
InBox) :- valid_proof(Prems, Goal, T2, [[RowNumb, Formula, assumption] | Proved],
1), valid_proof(Prems, Goal, T1, [[[RowNumb, Formula, assumption] | T2] | Proved],
InBox).

% lem
valid_proof(Prems, Goal, [[RowNumb, or(X, neg(X)), lem] | T], Proved, InBox) :-
valid_proof(Prems, Goal, T, [[RowNumb, or(X, neg(X)), lem] | Proved], InBox).

% contel
valid_proof(Prems, Goal, [[RowNumb, Formula, contel(X)] | T], Proved, InBox) :-
member([X, cont,_], Proved), valid_proof(Prems, Goal, T, [[RowNumb, Formula,
contel(X)] | Proved], InBox).

% andint
valid_proof(Prems, Goal, [[RowNumb, and(Xval, Yval), andint(X, Y)] | T], Proved,
InBox) :- member([X, Xval,_], Proved), member([Y, Yval,_], Proved),
valid_proof(Prems, Goal, T, [[RowNumb, and(Xval, Yval), andint(X, Y)] | Proved],
InBox).

% andel1
valid_proof(Prems, Goal, [[RowNumb, Formula, andel1(X)] | T], Proved, InBox) :-
member([X, and(Formula,_),_], Proved), valid_proof(Prems, Goal, T, [[RowNumb,
Formula, andel1(X)] | Proved], InBox).

% andel2
valid_proof(Prems, Goal, [[RowNumb, Formula, andel2(X)] | T], Proved, InBox) :-
member([X, and(_,Formula),_], Proved), valid_proof(Prems, Goal, T, [[RowNumb,
Formula, andel2(X)] | Proved], InBox).

% orint1
valid_proof(Prems, Goal, [[RowNumb, or(Xval, Yval), orint1(X)] | T], Proved, InBox)
:- member([X, Xval,_], Proved), valid_proof(Prems, Goal, T, [[RowNumb, or(Xval,
Yval), orint1(X)] | Proved], InBox).

% orint2
valid_proof(Prems, Goal, [[RowNumb, or(Xval, Yval), orint2(Y)] | T], Proved, InBox)
:- member([Y, Yval,_], Proved), valid_proof(Prems, Goal, T, [[RowNumb, or(Xval,
Yval), orint2(Y)] | Proved], InBox).

% impel
valid_proof(Prems, Goal, [[RowNumb, Formula, impel(X,Y)] | T], Proved, InBox) :-
member([X, Xval, _], Proved), member([Y, imp(Xval, Formula), _], Proved),
valid_proof(Prems, Goal, T, [[RowNumb, Formula, impel(X,Y)]|Proved], InBox).

% impint
valid_proof(Prems, Goal, [[RowNumb, imp(Xval, Yval), impint(X,Y)] | T], Proved,
InBox) :- getBox(X, Proved, NewList), member([X, Xval, _], NewList), member([Y,
Yval, _], NewList), valid_proof(Prems, Goal, T, [[RowNumb, imp(Xval, Yval),
impint(X,Y)] | Proved], InBox).

% negnegel
valid_proof(Prems, Goal, [[RowNumb, Formula, negnegel(X)] | T], Proved, InBox) :-
member([X, neg(neg(Formula)),_], Proved), valid_proof(Prems, Goal, T, [[RowNumb,
Formula, negnegel(X)] | Proved], InBox).

% negnegint
valid_proof(Prems, Goal, [[RowNumb, neg(neg(Formula)), negnegint(X)] | T], Proved,
InBox) :- member([X, Formula,_], Proved), valid_proof(Prems, Goal, T, [[RowNumb,
neg(neg(Formula)), negnegint(X)] | Proved], InBox).
```

DD1351 LAB2 2018
DIEGO LEON 961206
VIKTOR MEYER 971203

```
% negel
valid_proof(Prems, Goal, [[RowNumb, cont, negel(X,Y)] | T], Proved, InBox) :-
member([X, Xval, _], Proved), member([Y, Yval, _], Proved), neg(Xval) = Yval,
valid_proof(Prems, Goal, T, [[RowNumb, cont, negel(X,Y)] | Proved], InBox).

% egint
valid_proof(Prems, Goal, [[RowNumb, neg(Xval), negint(X,Y)] | T], Proved, InBox) :-
getBox(X, Proved, NewList), member([X, Xval, assumption], NewList), member([Y,
cont, _], NewList), valid_proof(Prems, Goal, T, [[RowNumb, Formula, negint(X,Y)] |
Proved], InBox).

% Copy
valid_proof(Prems, Goal, [[RowNumb, Formula, copy(X)] | T], Proved, InBox) :-
member([X, Formula, _], Proved), valid_proof(Prems, Goal, T, [[RowNumb, Formula,
copy(X)] | Proved], InBox).

% orel
valid_proof(Prems, Goal, [[RowNumb, Zval, orel(X, Y, Z, R, W)] | T], Proved, InBox)
:- getBox(Y, Proved, NewListY), getBox(R, Proved, NewListR), member([Y, Yval, _],
NewListY), member([R, Rval, _], NewListX), member([X, or(Yval, Rval), _], Proved),
member([Z, Zval, _], NewListY), member([W, Zval, _], NewListR), valid_proof(Prems,
Goal, T, [[RowNumb, Zval, orel(X, Y, Z, R,W)] | Proved], InBox).

% PBC
valid_proof(Prems, Goal, [[RowNumb, Formula, pbc(X,Y)] | T], Proved, InBox) :-
getBox(X, Proved, NewList), member([X, neg(Formula), assumption], NewList),
member([Y, cont, _], NewList), valid_proof(Prems, Goal, T, [[RowNumb, Formula,
pbc(X,Y)]|Proved], InBox).

% mt
valid_proof(Prems, Goal, [[RowNumb, neg(Xval), mt(X,Y)] | T], Proved, InBox) :-
member([X, imp(Xval, Yval), _], Proved), member([Y, neg(Yval), _], Proved),
valid_proof(Prems, Goal, T, [[RowNumb, neg(Xval), mt(X,Y)]|Proved], InBox).

% a valid proof cant end on an assumpiton
valid_proof(Prems, Goal, [], [[_, _, assumption]|T], 0) :- !, fail.

% a valid proof must end in its goal
valid_proof(Prems, Goal, [], [[_, Goal, _]|T], 0).

% we are currently in an assumption box
valid_proof(Prems, Goal, [], L, 1).

getBox(X, [L|_], L):- member([X, _, _], L).
getBox(X, [_|T], L):- getBox(X, T, L).
```