DD1350 Logik för dataloger
Laboration 1: Beviskontroll med Prolog

Mattias Cederlund
5/11 – 2013

Introduktion

Denna laboration har gått ut på att konstruera en beviskontrollerare för naturlig deduktion i Prolog. Som indata till programmet ges en sekvent och ett bevis i ND och programmet ska svara yes/no beroende om beviset är giltigt för given sekvent.

Metod

Algoritmen för beviskontrolleringen utgår från bevisets första rad och utvärderar beviset rad för rad, uppifrån och ner tills att den sista raden nås. Alla rader som redan kontrollerats sparas i en lista för att kontrollera att senare delar av beviset endast refererar till tidigare giltiga rader.

Om beviskontrolleraren redan validerat alla rader och den senaste validerade raden innehåller given sekvents slutsats så terminerar programmet med resultatet att beviset är giltigt för given sekvent. Detta förutsätter självklart att alla tidigare rader i beviset också är giltiga. Därmed kommer ett felaktigt bevis med rätt sista rad inte ge resultatet att hela beviset är sant.

Då det fortfarande finns rader i beviset som inte blivit validerade undersöks om den översta av de ej validerade raderna är en box genom att matcha det första elementet i raden med en rad innehållande ett antagande. Är så fallet är raden en box och man går vidare med att försöka validera boxen som ett eget bevis i sig genom predikatet valid_box. Sedan fortsätter valideringen med resterande rader i beviset.

Valideringen av boxar går till på precis samma sätt som validering av bevis utan box med undantaget att sista raden i en box inte behöver innehålla sekventens resultat. Hittas slutet av boxen terminerar boxvalideringen som sann och boxen läggs till i listan med validerade rader. Hittas en ny box valideras den före resterande del av den omslutande boxen.

Alla andra rader antas vara resultatet av en regel applicerad på tidigare validerade rader i beviset. När en sådan rad påträffas anropas valid_rule som består av ett predikat för varje giltig regel i ND samt ett för fallet då raden är en premiss. Till valid_rule skickas listan av redan validerade rader vilka ligger till grund för validering med hjälp av reglerna.

Om raden är en premiss undersöks om radens innehåll finns i listan med premisser och är så fallet är predikatet sant. Detta görs via den inbyggda funktionen member på listan med premisser. Är raden ett resultat av en regel hämtas regelns referenser ut och man undersöker om dessa rader innehåller det ursprungliga värdet före regelappliceringen, man så att säga applicerar regeln baklänges. Även denna undersökning görs genom den inbyggda funktionen member men här på listan med validerade rader.

Vissa regler refererar till boxar. I de fallen anropas find_box som letar i listan av verifierade rader efter den box som innehåller en viss rad. När rätt box påträffas returneras den. Denna funktion kan inte hitta nästlade boxar, men eftersom alla boxar valideras var för sig som egna bevis och man inte kan referera till rader i redan stängda boxar så är nästlade boxar inget vi behöver ta hänsyn till. När rätt box hittats görs kontrollen av regelappliceringen med hjälp av lämpliga rader i boxen.

Resultat

Vid körning av programmet terminerar både de två egna bevisen och alla fördefinierade testfall med förväntat resultat.

Predikat

Predikat	Sant	Falskt
Verify/1	När hela beviset verifierats.	
valid_proof/3	När hela beviset verifierats.	
valid_proof/4	Om det inte finns några ej validerade bevisrader och den senast validerade bevisraden innehåller sekventens slutsats.	
	Om lådan på första plats i listan av ej validerade bevisrader blivit validerad och resterande del av beviset blivit validerat.	
	Om bevisraden på första plats i listan av ej validerade bevisrader blivit validerad och resterande del av beviset blivit validerat.	Annars falskt.
valid_rule/3	Om given rad är resultatet av någon av de 15 ND-reglerna applicerad på någon validerad rad eller om raden är en premiss som finns i listan av premisser.	Annars falskt.
valid_box/4	Om det inte finns några ej validerade bevisrader i lådan.	
	Om lådan på första plats i listan av ej validerade bevisrader blivit validerad och resterande del av lådan blivit validerad.	
	Om bevisraden på första plats i listan av ej validerade bevisrader blivit validerad och resterande del av lådan blivit validerad.	Annars falskt.
find_box/3	Om bevisraden på första plats i listan av validerade bevisrader innehåller eftersökt rad.	
	Om svansen av listan av validerade bevisrader innehåller eftersökt rad.	Annars falskt.

Appendix A – Egna bevis

Valid

```
[neg(p)].

imp(p,q).

[
     [1, neg(p), premise],
     [
          [2, p, assumption],
          [3, cont, negel(2,1)],
          [4, q, contel(3)]
     ],
     [5, imp(p,q), impint(2,4)]
].
```

Invalid

```
[neg(and(p,q)), or(p,r), imp(neg(q),s)].

or(r,s).

[
     [1, neg(and(p,q)), premise],
     [2, or(p,r), premise],
     [3, imp(neg(q),s), premise],
     [
          [4, neg(s), assumption],
          [5, neg(q), mt(3,4)],
          [6, s, impel(5,3)],
          [7, cont, negel(4,6)]
     ],
     [8, s, contel(7)],
     [9, or(r,s), orint1(8)]
].
```

Appendix B – Programkod

```
verify(InputFileName) :-
        see(InputFileName),
        read(Prems), read(Goal), read(Proof),
        seen,
        valid_proof(Prems, Goal, Proof).

valid_proof(Prems, Goal, Proof) :-
        valid_proof(Prems, Goal, Proof, []).

%Om inga ovaliderade rader och senast validerade raden är slutsatsen.
valid_proof(_Prems, Goal, [], [[_Row, Goal, _Rule]|_Validate]).

%Första raden är en box som (måste) börjar med ett antagande.
valid_proof(Prems, Goal, [[[Row, Result, assumption]|Boxtail]|Prooftail], Validated) :-
        valid_box(Prems, Goal, Boxtail, [[Row, Result, assumption]|Validated]),
        valid_proof(Prems, Goal, Prooftail, [[[Row, Result, assumption]|Boxtail]|Validated]).

%Första raden är resultatet av en regel applicerad på tidigare validerad bevisrad.
valid_proof(Prems, Goal, [Proofhead|Prooftail], Validated) :-
        valid_rule(Prems, Proofhead, Validated),
        valid_proof(Prems, Goal, Prooftail, [Proofhead|Validated]).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%% RULES %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%premise
valid_rule(Prems, [_Row, Result, premise], _Validated) :-
        member(Result, Prems).

%negnegel(X)
valid_rule(_Prems, [_Row, Result, negnegel(X)], Validated) :-
        member([X, neg(neg(Result)), _], Validated).

%impel(X,Y)
valid_rule(_Prems, [_Row, Result, impel(X,Y)], Validated) :-
        member([X, Impprem, _], Validated),
        member([Y, imp(Impprem, Result), _], Validated).

%copy(X)
```

```
valid_rule(_Prems, [_Row, Result, copy(X)], Validated) :-
        member([X, Result, _], Validated).

%andint(X,Y)
valid_rule(_Prems, [_Row, and(And1, And2), andint(X,Y)], Validated) :-
        member([X, And1, _], Validated),
        member([Y, And2, _], Validated).

%andel1(X)
valid_rule(_Prems, [_Row, Result, andel1(X)], Validated) :-
        member([X, and(Result, _), _], Validated).

%andel2(X)
valid_rule(_Prems, [_Row, Result, andel2(X)], Validated) :-
        member([X, and(_, Result), _], Validated).

%contel(X)
valid_rule(_Prems, [_Row, _Result, contel(X)], Validated) :-
        member([X, cont, _], Validated).

%negnegint(X)
valid_rule(_Prems, [_Row, neg(neg(Result)), negnegint(X)], Validated) :-
        member([X, Result, _], Validated).

%orint1(X)
valid_rule(_Prems, [_Row, or(Result, _Other), orint1(X)], Validated) :-
        member([X, Result, _], Validated).

%orint2(X)
valid_rule(_Prems, [_Row, or(_Other, Result), orint2(X)], Validated) :-
        member([X, Result, _], Validated).

%lem
valid_rule(_Prems, [_Row, or(X, neg(X)), lem], _Validated).

%negel(X,Y)
valid_rule(_Prems, [_Row, cont, negel(X,Y)], Validated) :-
        member([X, Cont, _], Validated),
        member([Y, neg(Cont), _], Validated).

%mt(X,Y)
```

```
valid_rule(_Prems, [_Row, neg(Result), mt(X,Y)], Validated) :-
        member([X, imp(Result, Second), _], Validated),
        member([Y, neg(Second), _], Validated).

%negint(X,Y)
valid_rule(_Prems, [_Row, neg(Result), negint(X,Y)], Validated) :-
        find_box(X, Validated, Prevbox),
        member([X, Result, assumption], Prevbox),
        member([Y, cont, _], Prevbox).

%pbc(X,Y)
valid_rule(_Prems, [_Row, Result, pbc(X,Y)], Validated) :-
        find_box(X, Validated, Prevbox),
        member([X, neg(Result), assumption], Prevbox),
        member([Y, cont, _], Prevbox).

%impint(X,Y)
valid_rule(_Prems, [_Row, imp(Impprem, Result), impint(X,Y)], Validated) :-
        find_box(X, Validated, Prevbox),
        member([X, Impprem, assumption], Prevbox),
        member([Y, Result, _], Prevbox).

%orel(X,Y,U,V,W)
valid_rule(_Prems, [_Row, Result, orel(X,Y,U,V,W)], Validated) :-
        find_box(Y, Validated, FirstBox),
        find_box(V, Validated, SecondBox),
        member([X, or(First, Second), _], Validated),
        member([Y, First, _], FirstBox),
        member([U, Result, _], FirstBox),
        member([V, Second, _], SecondBox),
        member([W, Result, _], SecondBox).




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%% BOXHANTERING %%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Alla rader i lådan är validerade.
valid_box(_Prems, _Goal, [], _Validated).

%Första raden är en box som (måste) börjar med ett antagande.
valid_box(Prems, Goal, [[[Row, Result, assumption]|Boxtail]|Prooftail], Validated) :-
        valid_box(Prems, Goal, Boxtail, [[Row, Result, assumption]|Validated]),
        valid_box(Prems, Goal, Prooftail, [[[Row, Result, assumption]|Boxtail]|Validated]).

%Första raden är resultatet av en regel applicerad på tidigare validerad bevisrad.
valid_box(Prems, Goal, [Proofhead|Prooftail], Validated) :-
        valid_rule(Prems, Proofhead, Validated),
        valid_box(Prems, Goal, Prooftail, [Proofhead|Validated]).

%Första raden är en box som innehåller eftersökt rad.
find_box(Searchfor, [Boxhead|_Validated], Boxhead) :-
        member([Searchfor, _, _], Boxhead).

%Om inte leta efter raden i svansen.
find_box(Searchfor, [_|Validated], _Box) :-
        find_box(Searchfor, Validated, _Box).
```