DD1350 Logik för dataloger
Laboration 1: Beviskontroll med Prolog

Jarl Silvén och Simon Hellberg

30 november 2016

Generell beskrivning av beviskontroll

Först körs en bit kod när beviskontrollen instansieras:

1. Läser ut texten med den givna koden som delar upp det i Prems, Goal och Proof.
2. Försäkring om att beviset inte börjar med ett antagande.
3. Vänder på beviset baklänges (men lämnar boxar intakta, dessa vänds individuellt när de hittas). Om beviset kontrolleras bakifrån finns alla tillåtna rader kvar för varje rad.
4. Kontroll av att första raden i det nya bakåtvända beviset är målet.

När detta är klart startar en rekursion på det bakåtvända beviset, tillsammans med dess låda, som för huvudbeviset är hela beviset. Varje rad kontrolleras på följande vis:

1. Se om nuvarande lådan är tom, i så fall är det klart.
2. Se om regeln är premiss, om så är fallet ska den finnas i listan med premisserna.
3. Se om regeln är ett antagande, om så ska den vara först i en låda. I så fall måste nuvarande lådan utan antagandet vara tomt.
4. Se om det är en godkänd regel, genom att granska om regeln finns med i modulen ”rules”. Gör den det anropas regeln med två extra parametrar, beviset utan nuvarande rad, och nuvarande rad. Varför dessa skiljs åt beskrivs i det felaktiga exemplet.
5. När rekursionen det sista steget antas det vara en låda. Är det inte en låda returneras false snabbt av andra skäl. Lådan vänds baklänges, och ny rekursion startas på resten av beviset tillsammans med lådans rader.

Efter varje steg anropas beviskontrollen rekursivt med svansen på baklängesbeviset, så att steg 2-5 upprepas tills sluttillståndet 1 nås.

Boxhantering

Hur lådor hanteras i huvudrekursionen har redan beskrivits, på tidigare punkt 5. Men lådor hanteras också av de regler som hänvisar till dem. Om lådan hänvisas av regel med index Xi,Yi, ska börja på A och resultera i C går det till så här:

1. Leta igenom bevis efter låda som börjar på index Xi.
2. När lådan hittats, kolla om antagandet är A.
3. Vänd på lådan bak och fram.
4. Kolla att lådan slutar på index Yi, och resulterar i konsekvensen C.

På detta vis kan regler referera till lådor endast om de referar till hela boxen, inte till individuella rader. Detta eftersom lådorna, innan de upptäckts av huvudrekursionen, inte ”konverterats” till rader än. De kan ej heller referera till djupare lådor - endast lådor på samma nivå.

Predikat-tabell

Predikaten är sanna IFF det som står under rubriken sann om gäller, annars är det falska.
Variablernas namn för tabell, med förklaringar:

- Xi = Första index.
- X = Värde vid X index.
- Yi = Andra index.
- Y = Värde vid y index
- P = "Proof", bevis. Det andra raderna som går att nå i beviset.
- R = Resultat, på nuvarande raden.
- A = Antagande, först i låda.
- C = Consequence, Resultat av låda.
- $\perp$ = Contradiction, motsägelse.
- Z = Vad som helst.

Lådor markeras Xi-Yi $\rightarrow$ A,C.

Predikat	Sant
copy(Xi,P,R)	Om R = X.
andint(Xi,Yi,P,R)	Om R = and(X,Y).
andel1(Xi,P,R)	Om X = and(R,Z).
andel2(Xi,P,R)	Om X = and(Z,R).
orint1(Xi,P,R)	Om R = or(X,Z).
orint2(Xi,P,R)	Om R = or(Z,X).
orel(Xi,Yi,Ui,Vi,Wi,P,R)	Om X = or(A_1,A_2) och R = C_1 och R = C_2, där Yi-Ui $\rightarrow$ A_1,C_1 och Vi-Wi $\rightarrow$ A_2,C_2.
impint(Xi,Yi,P,R)	Om R = imp(A,C), där Xi,Yi $\rightarrow$ A,C.
impel(Xi,Yi,P,R)	Om Y = imp(X,R).
negint(Xi,Yi,P,R)	Om R = neg(A) och C = $\perp$, där Xi-Yi $\rightarrow$ A,C.
negel(Xi,Yi,P,R)	Om Y = neg(X) och R = $\perp$.
contel(Xi,P,R)	Om X = $\perp$.
negnegint(Xi,P,R)	Om R = neg(neg(X)).
negnegel(Xi,P,R)	Om X = neg(neg(R)).
mt(Xi,Yi,P,R)	Om X = $Z_1 \rightarrow Z_2$ och Y = neg(Z_2) och R = neg(Z_1)
pbc(Xi,Yi,P,R)	Om R = neg(A) och C = $\perp$, där Xi-Yi $\rightarrow$ A,C.
lem(P,R)	Om R = or(Z,neg(Z)) eller om R = or(neg(Z),Z).

Ett exempel - korrekt bevis

I ett exempelvis ska vi bevisa sekventen $(p \vee q) \rightarrow q, \neg q \vdash \neg p$ med nedanstående bevis. Bevisalgoritmen arbetar från slutet till början, och använder reglerna definierade ovan.

Exempelbeviset inkluderade ej lådor djupare än en nivå, men rekursionen hanterar alla lådor likvärdigt och alla givna testfall avklarades.

1	$(p \vee q) \rightarrow q$	premise
2	$\neg q$	premise
3	$\neg(p \vee q)$	MT 2,1
4	p	assumption
5	$p \vee q$	or_{i_1} 4
6	$\bot$	$\neg_e$ 5,3
7	$\neg q$	copy 2
8	$\neg p$	assumption
9	$\neg p$	$\neg_i$ 4-6

Rad 9 är okej eftersom $\neg p$ är målet, och att boxen 4-6 har p som antagande och $\bot$ som slutsats, vilket möjliggör $\neg_i$

1	$(p \vee q) \rightarrow q$	premise
2	$\neg q$	premise
3	$\neg(p \vee q)$	MT 2,1
4	p	assumption
5	$p \vee q$	or_{i_1} 4
6	$\bot$	$\neg_e$ 5,3
7	$\neg q$	copy 2
8	$\neg p$	assumption
	$\vdots$	

Rad 8 är assumption i början av en box och därför okej, även om boxen inte används.

1	$(p \vee q) \rightarrow q$	premise
2	$\neg q$	premise
3	$\neg(p \vee q)$	MT 2,1
4	p	assumption
5	$p \vee q$	or_{i_1} 4
6	$\bot$	$\neg_e$ 5,3
7	$\neg q$	copy 2
	$\vdots$	

Rad 7 kopierar en rad rätt och är därför okej, även om den aldrig används.

1	$(p \vee q) \rightarrow q$	premise
2	$\neg q$	premise
3	$\neg(p \vee q)$	MT 2,1
4	p	assumption
5	$p \vee q$	or_{i_1} 4
6	$\bot$	$\neg_e$ 5,3
	$\vdots$	

Rad 6 är rätt därför att två motsatta rader tidigare i beviset resulterar i en motsägelse.

1	$(p \vee q) \to q$	premise
2	$\neg q$	premise
3	$\neg(p \vee q)$	MT 2,1
4	p	assumption
5	$p \vee q$	or_{i_1} 4
	$\vdots$	

Rad 5 är rätt eftersom en av påståendena finns ovan i beviset, vilket räcker för or introduktion

1	$(p \vee q) \to q$	premise
2	$\neg q$	premise
3	$\neg(p \vee q)$	MT 2,1
4	p	assumption
	$\vdots$	

Rad 4 är rätt därför att en box öppnats med ett assumption

1	$(p \vee q) \to q$	premise
2	$\neg q$	premise
3	$\neg(p \vee q)$	MT 2,1
	$\vdots$	

Rad 3 är rätt därför att tillräckligt med information för MT finns tidigare i beviset

1	$(p \vee q) \to q$	premise
2	$\neg q$	premise
	$\vdots$	

Rad 2 är rätt därför att påståendet finns med bland premisserna

1	$(p \vee q) \to q$	premise
	$\vdots$	

Rad 1 är rätt därför att påståendet finns med bland premisserna

Ett till exempel - felaktigt bevis

I detta exempel ska den triviala sekventen $p \vdash p$ bevisas med så många felaktigheter som möjligt, som påträffats under arbetets gång men som lösts och nu inte längre går igenom.

1	p	assumption
2	$p \vee \neg p$	LEM
3	$\neg p$	get_seq 1
4	p	copy 4
5	$\bot$	$\neg_e$ 3,4
6	p	PBC 3-5

På rad 6 försöker PBC hämta information från rader 3-5, men dessa motsvarar inte början/slut på box.

1	p	assumption
2	$p \vee \neg p$	LEM
3	$\neg p$	get_seq 1
4	p	copy 4
5	$\bot$	$\neg_e$ 3,4
	$\vdots$	

Rad 5 är den enda i det felaktiga beviset som är rätt.

1	p	assumption
2	$p \vee \neg p$	LEM
3	$\neg p$	get_seq 1
4	p	copy 4
	$\vdots$	

På rad 4 försöker copy kopiera sig själv. Detta undveks genom att regler inte får se sig själva.

1	p	assumption
2	$p \vee \neg p$	LEM
3	$\neg p$	get_seq 1
	$\vdots$	

På rad 3 används en hjälpfunktion. Att otillåtna regler används hindras av att endast predikat från regelmodulen tillåts.

1	p	assumption
2	$p \vee \neg p$	LEM
	$\vdots$	

Regeln på rad 2 är rätt, men en box ska börja med ett antagande. Detta kollas när box upptäcks.

1	p	assumption
	$\vdots$	

I huvudbeviset får inga antaganden finnas med, inte ens i början.

Appendix A - Källkod till huvudfilen

```
:- use_module(rules).

% verify that the file contains a valid proof
verify(InputFileName) :- see(InputFileName),
        read(Prems), read(Goal), read(Proof),
        seen,
        valid_proof(Prems, Goal, Proof).

% the main proof box cannot start with an assumption
valid_proof(_,_,[[_,_,assumption]|_]) :- !,fail.
% make sure a logical proof is syntactically correct
valid_proof(Prems, Goal, Proof) :-
        reverse(Proof,Foorp),
        Foorp = [[_,Goal|_]|_],
        prove(Prems,Foorp,Foorp).

% prove one row, P = premises, F = foorp, C = current box, K = connective
prove(_,_,[]) :- !.
% introducing rows from premises, only predicate that uses P
prove(P,[_|FT],C) :- C = [[_,S,K]|CT], K = premise,!,
        in_list(S,P), prove(P,FT,CT).
% assumption, only possible at the start of a box
prove(P,[_|FT],C) :- C = [[_,_,K]|CT], K = assumption,!,
        CT = [], prove(P,FT,CT).
% rules, each take its result and the tail of the proof as extra parameters
prove(P,F,[_|CT]) :- F = [[_,R,K]|FT],
        catch((functor(K,N,A),B is A + 2,functor(L,N,B)),error(_,_),fail),
        predicate_property(L,imported_from(rules)), % check that predicate
        call(K,FT,R),prove(P,FT,CT),!.
% next row must be a box, B = box, X = xob, Y = rest of proof including box
prove(P,[B|F],[B|C]) :- B = [[_,_,assumption]|_], reverse(B,X), append(X,F
        prove(P,Y,X), prove(P,F,C).

% check if element is in list ( because member ruins testing.. )
in_list(_,[]) :- fail.
in_list(X,[H|_]) :- X = H,!.
in_list(X,[_|T]) :- in_list(X,T).
```

Appendix B - Källkod till Reglerna

```
:- module(rules,[
                copy/3,andint/4,andel1/3,andel2/3,
                orint1/3,orint2/3,orel/7,impint/4,impel/4,
                negint/4,negel/4,contel/3,negnegint/3,negnegel/3,
                mt/4,pbc/4,lem/2
        ]).

% General variable naming pattern:
% Xi = index x, X = value at x, Yi = index y, Y = value at y,
% P = proof, R = result, C = Result of box, A = Assumption.

% copy a previous row.
copy(Xi,P,R) :-
        get_seq(Xi,P,X),
        R = X.

% and introduction from Xi and Yi
andint(Xi,Yi,P,R) :-
        get_seq(Xi,P,X),
        get_seq(Yi,P,Y),
        R = and(X,Y).

% and elimination at xi first element.
andel1(Xi,P,R) :-
        get_seq(Xi,P,and(R,_)).

% and elimination at xi second element.
andel2(Xi,P,R) :-
        get_seq(Xi,P,and(_,R)).

% or introduction at xi first element.
orint1(Xi,P,R) :-
        get_seq(Xi,P,X),
        R = or(X,_).

% or introduction at xi second element.
orint2(Xi,P,R) :-
        get_seq(Xi,P,X),
        R = or(X,_).

% or elimination from Xi. or: Xi  box1: Yi-Ui box2: Vi-Wi, A1-2= Assumptions
orel(Xi,Yi,Ui,Vi,Wi,P,R) :-
        get_seq(Xi,P,X),
        get_box(Yi,Ui,P,A1,R),
        get_box(Vi,Wi,P,A2,R),
        X = or(A1,A2).

```

```
% implication introduction
impint(Xi,Yi,P,R) :-
        get_box(Xi,Yi,P,A,C),
        R = imp(A,C).

% removes implication.
impel(Xi,Yi,P,R) :-
        get_seq(Xi,P,X),
        get_seq(Yi,P,Y),
        Y = imp(X,R).

% negation introduction.
negint(Xi,Yi,P,R) :-
        get_box(Xi,Yi,P,A,cont),
        R = neg(A).

% negation elimination.
negel(Xi,Yi,P,R) :-
        get_seq(Xi,P,X),
        get_seq(Yi,P,neg(X)),
        R = cont.

% contradiction elimination.
contel(Xi,P,_R) :-
        get_seq(Xi,P,cont).

% introduces doublenegation.
negnegint(Xi,P,R) :-
        get_seq(Xi,P,X),
        R = neg(neg(X)).

% eliminates double negation.
negnegel(Xi,P,R) :-
        get_seq(Xi,P,neg(neg(R))).

% MT, modus tolens, x->y if y is false x cannot be true.
mt(Xi,Yi,P,R):-
        get_seq(Xi,P,imp(A,B)),
        get_seq(Yi,P,neg(B)),
        R= neg(A).

%Proof by contradiction, if neg(x) leads to contradiction x must be true.
pbc(Xi,Yi,P,R) :-
        get_box(Xi,Yi,P,A,cont),
        A = neg(R).

%Law of excluded middle, X or Neg(X) must be true.
lem(_P,R) :-
        R = or(X,neg(X)).
```

```
97 lem(_P,R) :-
98         R = or(neg(X),X).
99
100 % find sequence at index, I = index, H = head, T = tail S = sequence.
101 get_seq(_,[],_) :- fail.
102 get_seq(I,[[H,S,_]|_],S) :- I = H,!.
103 get_seq(I,[_|T],S) :- get_seq(I,T,S).
104
105 % find correct box, Bi = begin, Ei = end, P = proof, A = assumption, C = result
106 get_box(_,_,[],_,_) :- fail.
107 get_box(Bi,Ei,[B|_],A,C) :- B = [[Bi,A,_]|_],reverse(B,[[Ei,C,_]|_]),!.
108 get_box(Bi,Ei,[_|T],A,C) :- get_box(Bi,Ei,T,A,C).
```