Noel Karlsson Johansson, noelkj@kth.se, 000519-0517
Richard Palm, ricpal@kth.se, 001111-5276

Labb 2

Generell beskrivning

Den första funktionen i vårt program som körs är valid_proof/3. Den tar in premisserna, målet för beviset och själva beviset som argument. Den kollar sedan att sista raden är samma som målet med verifyLastLine/2 och kör verifyProof/3 som går igenom beviset rad för rad och verifierar dem med verifyLine/3 functionen. verifyLine använder pattern matching för att kunna verifiera varje rad. Först är radnummret, sen är det själva termen som bevisas på raden och sist är vilken regel som använts. Pattern matching används för att ta reda på vilken regel som används och sen används olika fall för varje regel för att se att den stämmer. Antagande behöver inte verifieras alls utan startar bara en ny box och premisser verifieras genom att jämföras med premisserna som skrevs in separat från beviset. Alla andra regler kollar vilken rad som regeln refererar till och ser om det den använder för själva operationen finns på den raden. VerifyLine kollar också samtidigt att reglerna bara refererar till rader den kommer åt med hjälp av compareBoxNumber och compareLevel.

Box Hantering

Assumption boxar hanteras som ett eget bevis, med hjälp av recursion. Det finns dock restriktioner när man refererar till en box och när man är i en box. En rad kan inte använda saker som är innanför en box som de själva inte är med i, samt att om man refererar till en box ska raden referera till hela boxen(alltså båda till slutet och början av boxen).

Ordlista

Rad, är den delen av Proof som representeras av: [LineNum, Term, Func].

Radnummer, är den första delen av en rad LineNum.

Term, är andra delen av en rad, är resultatet av applicerandet av en lag, premiss eller assumption.

Funktion, hur en term på en rad kom till.

Box, Startar med en assumption, kan ses som ett litet bevis innanför beviset.

Refererar, tex en funktion impel(1,2) refererar till raden 1 och 2.

Predikat (tabell)

Predikat	Sann	Falsk
valid_proof	När beviset fungerar och sista raden stämmer överens med Goal	När beviset inte fungerar eller sista raden inte stämmer överens med Goal
verifyProof	När hela beviser är sant	Om beviset inte stämmer
verifyLastLine	När sista linjen stämmer överens med goal	När sista linjen inte stämmer överens med goal
checkLastLine	När en rad har samma term som den term man jämför med	när radens term och andra argumentets term inte är samma
verifyLine(premiss)	Om en rad är en premiss och uttrycket finns i premissen	Om en rad är en premiss och uttrycket inte finns i premissen
verifyLine(assumption)	om beviset innanför boxen är san	om beviset innanför boxen inte stämmer
verifyLine(term)	Om en funktion använder kända regler på rätt sätt	Om den inte använder en känd regel eller en känd regel fel
findTerm	Om den hittar en viss term på en viss rad	Om den inte hittar termen i argumentet på en viss rad eller om raden inte finns.
boxLevel	Kollar hur många boxar in en rad är. (Tex har du ett bevis med bara en box, så är en rad där på level = 1) är sann så länge strukturen på input är korrekt	Om en rad inte är i en box
countBoxes	Räknar det totala antalet lådor som används i beviset.	Falsk om beviset är strukturerat fel.
detBox	Sann om countBoxes, elementComp och boxLevel är sann eller boxLevel returnerar level = 0.	Falskt om någon av predikaten returnerar falskt(vilket inte borde hända).

elementComp	Om ett nummer är mellan två andra nummer i en lista av nummer.	Om ett nummer inte är mellan två andra nummer i en lista av nummer.
compareBoxNumber	Är sann om en rads funktion bara refererar till en hel box och inte till delar av en box.	När en term refererar till en box som den inte ska kunna
boxNumberHelper	-||-	-||-
compareLevel	Är sann om två rader är tillräckligt nära varandra i level. Hur nära andras beroende på vilken funktion som raden använder.	När rader är för långt ifrån varandra d.v.s för många boxar mellan dem.

Predikatet verifyLine

Predikatet verifyLine varierar lite beroende på vad det är för funktion som ska verifieras. Men vissa saker stämmer för alla som att en funktion kan inte referera till en rad som händer efter den i sekvensen. Detta kollas via en simpel ‘X < LineNum’ i verifyLine.

Alla nästan alla funktionen referar till någon annan rad. Här används findTerm för att hitta vad som står på en rad. Den kommer användas lite olika beroend på vilken funktionen är men den används för att verifiera att rätt saker står på raden som refereras.

Ett annat predikat som används mycket i verifyLine är compareLevel och compareBoxNumber, båda dessa används för att verifiera att att en funktion inte referar till en rad som är i en box. ‘Level’ i detta sammanhang betyder hur långt in i en box en rad är. Tex om du har ett Proof men bara en assumption och en rad befinner sig i den boxen då är du på level = 1. Med boxnummer menas i vilken box från toppen till botten du är i. Ta förra exemplet, i det fallet skulle du om du var i boxen befinna dig i boxnummer = 1.

Alla dessa tillsammans utgör alla villkor för att predikatet veryLine ska vara sant.

- copy(X)
    - X < Radnummer
    - På rad X finns samma term som på raden med copy
    - Referar bara till saker som inte är i en box eller är i samma box.
- andint(X,Y)

    - X,Y < Radnummer
    - X och Y refererar bara till saker utanför en box eller i samma box.
    - På rad X finns första delen av and(A,B) och andra delen finns på rad Y
    - Referar bara till saker som inte är i en box eller är i samma box.
- andel1(X)/andel2(X)
    - X < Radnummer
    - På rad X finns det ett and(A,B) och där är A/B = Term.
- orint1(X)/orint2(X)
    - X < Radnummer
    - På rad X finns or(A,B) och A/B = Term.
    - Referar bara till saker som inte är i en box eller är i samma box.
- orel(X,Y,U,V,W)
    - X,Y,U,V,W < Radnummer
    - På rad X finns det or(A,B) och på rad Y finns det en term = A och på rad V en term = B. Termen på samma rad som funktionen ska finnas på rad U och W samt att (Y och U)/(V och W) är i samma box, samt att Y och V refererar till starten av en box och U och W refererar till slutet.
- impint(X,Y)
    - X,Y < Radnummer
    - X och Y är i samma box samt X är start och Y är slutet på boxen samt att termen på raden med funktionen är imp(A,B) och termen på raden X,Y = A,B.
- impel(X,Y)
    - X,Y < Radnummer
    - X och Y är inte för många levels ifrån och på rad X finns bevis för A och på rad Y finns det imp(A,B), då kan B vara termen som bryts ut.
- contel(X)
    - X < Radnummer
    - på rad X ska det finnas en motsägelse
- negint(X,Y)
    - X,Y < Radnummer
    - X ska vara numret på en rad där termen A antas. Y ska vara radnumret i slutet av boxen där A antas på första raden och ska innehålla cont. Då kan termen vara neg(A)
- negel(X,Y)
    - X,Y < Radnummer
    - X är radnummret där A är bevisat, Y är radnumret där neg(A) är bevisat. De ska inte vara för många levlar infrån och termen ska vara en motsägelse(cont)
- negnegint(X)
    - X < Radnummer
    - Om A är bevisat på rad X och den inte är för många levels ifrån så kan termen vara neg(neg(A))
- negnegel

    - X < Radnummer
    - Om neg(neg(A)) är bevisat på rad X och den inte är för många levels ifrån så kan termen vara A
- mt(X,Y)
    - X,Y < Radnummer
    - Om det på rad X finns en implikation imp(A,B) och på rad Y finns neg(B) så kan termen vara neg(A)
- pbc(X,Y)
    - X,Y < Radnummer
    - X ska vara numret på en rad där termen neg(A) antas. Y ska vara radnumret i slutet av boxen där neg(A) antas på första raden och ska innehålla cont, då kan termen vara A.
- lem
    - pattern matcher så att det antingen står or(neg(A), A) eller or(A, neg(A)) i termen.

All dessa är regler med två variabler som där variablerna refererar till två olika rader

Appendix

```
[q].

and(imp(p,q), imp(c,q)).

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

```
[q].

and(imp(p,q), imp(c,q)).

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