Kort populärvetenskaplig sammanfattning av min doktorsavhandling*

Jakob Nordström†

September 2009

Min forskning handlar om att förstå hur svårt det är att bevisa att saker är sanna. Varför är det intressant?

Jo, ett skäl är att moderna mjukvarusystem (dvs. datorprogram) och hårdvarusystem (t.ex. processorer inuti datorerna) har blivit allt mer komplexa de senaste decennierna allteftersom tekniken utvecklats mer och mer. I datorernas barndom var systemen så pass små att en ingenjör som tänkte tillräckligt djupt och tillräckligt länge kunde sätta sig in i ett helt system och förstå varje detalj av hur det fungerade. Det går inte längre, eftersom konstruktionerna är alldeles för stora. Istället försöker man simulera nya system innan man bygger dem, och sedan testa dem när man har byggt klart för att kontrollera om allt blev rätt.

Problemet är att systemen är så oerhört stora och invecklade att inte heller simulering och testning räcker till för att täcka in alla fall. Samtidigt kan oupptäckta fel få förödande konsekvenser.

Idag finns datorer precis överallt. I många fall kan vi ha överseende med att det blir lite fel – till exempel kan vi kanske acceptera att datorn "hänger sig" då och då när vi sitter hemma vid skrivbordet och arbetar med ordbehandling eller kalkylblad. Men om datorn styr ABS-bromsarna i bilen så får det bara inte bli fel när de behövs. Om datorn reglerar en pacemaker så måste den fungera hela tiden. Och det är inte acceptabelt om datorn hänger sig ifall det råkar vara datorn i autopiloten på en jumbojet på 10 000 meters höjd... Men tyvärr blir det ibland fel även i sådana system.

Vad kan man göra åt detta? Det räcker inte med att testa och se att det *verkar* fungera – vi skulle vilja *veta helt säkert* att det inte finns några fel.

Som tur är finns det en vetenskap som sysslar med helt säker kunskap, nämligen matematiken. I matematikens underbara värld är allting svart eller vitt, sant eller falskt. Så ett sätt att angripa problemet är att göra exakta matematiska beskrivningar av hur vi vill att våra tekniska system ska fungera, och beskriva föreslagna designlösningar på samma sätt. Då kan man sedan använda matematiska verktyg för att *bevisa* att konstruktionslösningarna är felfria.

När man gör sådana matematiska beskrivningar får man som resultat en stor formel som beskriver systemet. Formeln har en massa variabler som svarar mot alla de olika tillstånd som systemet kan befinna sig i, och om formeln alltid är sann oavsett vad variablerna har för värden så svarar det mot att designen är korrekt.

*Det är inte helt okomplicerat att försöka sammanfatta fem års forskning på knappt två sidor. Det säger sig självt att det som står här är ganska förenklat, och på sina ställen är det så pass generaliserat att det till och med är lite fel. Men förhoppningsvis kan detta ändå ge en känsla för vad det hela handlar om.

†Adress: Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA. E-post: jakobn@mit.edu.

De här formlerna kan vara ohyggligt stora med miljontals variabler, så det är ingenting som man löser med penna och papper. Istället använder man datorer för beräkningarna. Därför är följande fråga högintressant:

Hur svårt är det att med dator bevisa (eller motbevisa) att formler är sanna?

Denna fråga har inte bara stor praktisk betydelse. Om vi kunde besvara frågan helt skulle vi också kunna lösa ett av de viktigaste teoretiska problemen inom den moderna matematiken och datavetenskapen. Ett av de sju berömda *millennieproblemen* som Clay Mathematics Institute satt upp som de stora utmaningarna inför det nya årtusendet (`www.claymath.org/millennium/`), och som kan göra den som löser det en miljon dollar rikare, handlar just om detta. (I teknisk jargong är det känt som problemet om huruvida P är lika med NP eller inte.)

De bästa programmen som finns idag för att bevisa logiska formler bygger på en metod som kallas *resolution*. För sådana program är det två frågor som har avgörande betydelse: (1) Hur snabbt går programmen? (2) Hur mycket datorminne behöver de?

Att tid är en viktig faktor är lätt att förstå – programmen måste gå fort för att vara användbara i praktiken. Men det är också viktigt hur mycket datorn behöver "hålla reda på samtidigt" när den letar efter bevis. Ju mindre minne som behövs, desto enklare blir det. Och det har visat sig att minnesanvändningen är den stora flaskhalsen för program som bygger på resolution. Så här finns en potentiell konflikt: vi vill att programmen ska gå så fort som möjligt, men vi vill också hålla nere minnesanvändningen. Går det att nå båda dessa mål på samma gång? Det vill säga: Om det finns ett kort bevis, kan vi vara säkra på att det då också går att hitta med lite minne?

De här är något som man började forska kring i slutet av 1990-talet, men trots att många forskare har varit aktiva inom området har frågorna förblivit olösta. I min doktorsavhandling på KTH, som jag följt upp i min fortsatta forskning på Massachusetts Institute of Technology, har jag givit uttömmande svar på dessa frågor.

Lite överraskande visar det sig att det *inte alls* är möjligt att hålla både tids- och minnesåtgången nere samtidigt. Tvärtom finns det formler som å ena sidan har väldigt korta bevis och å andra sidan har väldigt minnessnåla bevis, men där alla korta bevis kräver stora mängder minne, och alla minnesnåla bevis är så orimligt långa att de inte går att hitta i praktiken. Detta betyder att den naturliga strategin att försöka begränsa minnesanvändningen när man söker efter bevis i vissa fall kan ha katastrofala konsekvenser.

Det är för denna forskning som jag har fått Ackermann Award 2009 för "enastående avhandlingar i datavetenskaplig logik" från European Association for Computer Science Logic.