Seminarie 4

Constraint networks

Testning

Enhetstesterna testar tilldelning till in och ut-anslutningarna för både Atter och Multiplier. Ett lite mer avancerat test med omvandling mellan celsius och fahrenheit körs även.

Vi började med att göra enhetstester för Adder och Multiplier. Vi upptäckte ganska fort att ArithmeticConstraint får problem att sätta in-värden när ut-värdet ändras, vilket vi fixade enligt nedan.

Kod

För att lösa problemet vi stötte på under testningen kollar vi om det är ut-anslutningen som har fått ett nytt värde, och om en ingång saknar värde. Då uppdaterar vi den utan till korrekt värde, med invers- operatorn.

Constraint parser

Kod

Vi har gjort fyra ändringar från ursprungliga koden.

Vi har uppdaterat matchningarna så båda sidorna av operationer kan vara av samma typ. Detta gör att t.ex. `a=b+2` är korrekt, vilket tidigare bara kunde skrivas utan parenteser som `b+2=a`, trots att `a=(b+2)` fungerade.

Vi skapade funktionen `get_connector(side)` som returnerar den `Connector` som ska användas för samansättning av ekvationsled och nästlade uttryck. Vår tankegång kring vilken anslutning som ska returneras började i `get_connectors(conn_a,op,conn_b)`, som används för att skapa den tredje anslutningen i operationer. Namnet som tilldelas denna är nämligen en summa av de andra två. Vi drog därför slutsatsen att den anslutning med längst namn alltid är resultatet av operationen, och därför den som ska returneras.

För att hantera nästlade uttryck uppdaterade vi `get_connectors(conn_a,op,conn_b)` att använda `get_connector(side)`. Detta säkerställer att ekvationer som `a+2=1*(b+1)` beräknas korrekt.

Den sista ändringen skedde i `replace_conn(lh, rh)`, där vi justerade vilken `Connector` i `ArithmeticConstraint` som ersattes. Detta görs då subtraktion och division ändrar anslutningen som bär resultatet. Här gjorde vi ett litet fusk

med `eval()` för att göra anslutningen på rätt in- eller utgång. Kanske ni har ett snyggare sätt att lösa det på?

Testning

För parsern har vi enhetstest för additions- och multiplikationsekvationer. Vi testar att ha uttryck på båda sidor av likheterna, och att subtraktion och division fungerar. Sedan testar vi några mer komplexa ekvationer för att säkerställa att nästlade uttryck och att prioriteringsregler följs.

Vi har ett problem. När ett constraint är anslutet med två `ConstantConnector`s fungerar nätverket inte som förväntat. Vi har ett par exempel i testfilen. Detta gör att vissa ekvationer med uttryck som endast innehåller konstanter inte går att lösas ut korrekt. Vi upptäckte detta ganska sent och har inte riktigt tid att lösa det.