Syntax

Wir implementieren Formeln (der Aussagenlogik) mithilfe eines neuen types, der als Atome einen flexiblen Typen hat. Z.B. können hier Terme implementiert werden und auch Signaturen.

Funktionen für Konstruktoren und Destruktoren: Aufbauen und Auseinandernehmen von Formeln. Es ist vorteilhaft die Konstruktoren als Funktion zu haben um sie anderen Funktionen übergeben zu können.

onatoms Funktion: Manipulieren der Atome einer Formel, z.B. Substituieren der Atome

overatoms Funktion: Entnehmen der Informationen, die in den Atomen enthalten sind, z.B. verwendet in atoms Funktion, die die Menge der Atome einer Formel (als Liste) herausgibt.

Semantik

Vorerst schauen wir uns propositionale Formeln an, d.h. ohne Quantoren und mit dem type Prop, der in den type der Formeln eingesetzt wird. Prop sollen bei einer Auswertung entweder wahr oder falsch annehmen.

eval Funktion wertet, für eine gegebene Auswertung der Atome einer propositionalen Formel, den Wahrheitswert aus. Dabei geschiet die Auswertung kanonisch mit den in Ocaml eingebauten logischen Operatoren. Auf ähnliche Weise haben wir den semantischen Wert einer Formel in der Ëinführung in die mathematische Logik"definiert.

Wahrheitstabellen

zwei Zwischenergebnisse:
1. die Menge der Atome in einer propositionalen Formel ist endlich
2. Sind zwei Auswertungen v, v' auf der Menge der Atome einer Formel p gleich, so ist auch eval v p = eval v' p
Damit sind alles Informationen einer propositionalen Formel in einer endlichen Wahrheitstabelle codiert.

print_truthtable ist die entsprechende Funktion, die für eine propositionale Formel die Wahrheitstabelle printed.

Tautologien

Falls jede Auswertung einer Formel wahr ergibt, nennen wir die Formel eine Tautologie. Die tautology Funktion überprüft diese Eigenschaft an einer Formel. Damit ist sie die erste Funktion, die uns eine semantische Aussage über Formeln treffen lässt.

Theorem: eval (psubst $(x \Rightarrow q)p)v$ = eval $p((x \mapsto \text{eval } q\ v)v)$,
für x Atom, p, q Formeln, v Auswertung, psubst Substitutionsfunktion.

Korollar: Sei p Tautologie, x Atom, q Formel, dann ist psubst($x \Rightarrow q)p$ eine Tautologie.

Theorem: Sei v Auswertung, p, q, r Formeln, s.d. eval $p\ v$ = eval qv,
dann eval (psubst $(x \Rightarrow p)r)v$ = eval (psubst $(x \Rightarrow q)r)v$

Korollar: falls $p \iff q$, dann eval (psubst $(x \Rightarrow p)r)v$ = eval (psubst $(x \Rightarrow q)r)v$
Dieses Resultat erlaubt es uns logisch äquivalente Formeln füreinander einzusetzen, ohne, dass sich das semantische Verhalten der Formel ändert.

Ein paar wichtige Tautologien:
$(p \wedge q) \iff (q \wedge p)$
$(p \wedge (q \wedge r)) \iff ((p \wedge q) \wedge r)$
$(p \wedge (q \vee r)) \iff ((p \wedge q) \vee (p \wedge r))$
$(p \vee (q \wedge r)) \iff ((p \vee q) \wedge (p \vee r))$
$(p \vee (q \iff r)) \iff (p \vee q \iff p \vee r)$
$(p \wedge q) \iff ((p \iff q) \iff p \vee q)$
$(p \implies q) \iff (\neg q \implies \neg p)$
$(p \iff (q \iff r)) \iff ((p \iff q) \iff r)$
$(p \iff q) \iff (q \iff p)$
De Morgan Laws:
$\neg(p \vee q) \iff (\neg p \wedge \neg q)$
$\neg(p \wedge q) \iff (\neg p \vee \neg q)$

Um uns in Richtung Normalformen zu bewegen, können wir Operatoren durch äquivalente Ausdrücke ersetzen (z.B. de Morgan Laws) und überlegen welche Operatoren ausreichen um alle anderen darzustellen.

Die Menge $\{\wedge, \neg\}$ stellt vollständig $\{\wedge, \neg, \vee, \Rightarrow, \Leftrightarrow, \top, \bot\}$ dar
$\bot \Leftrightarrow p \wedge \neg p$
$\top \Leftrightarrow \neg(p \wedge \neg p)$
$p \vee q \Leftrightarrow \neg(\neg \wedge \neg q)$
$p \Rightarrow q \Leftrightarrow \neg(p \wedge \neg q)$
$p \Leftrightarrow \neg(p \wedge \neg q) \vee \neg(\neg p \wedge q)$