Handout
Normalforms

von Lars Nießen

Wintersemester 2023/24

30. Oktober 2023

1 Motivation

Eine Formel kann eine Anfangs eher sehr unübersichtliche Struktur haben um sie selbst direkt auf korrektheit zu überprüfen. Unser Ziel ist es zu einer beliebigen Formel eine logisch äquivalente Formel zu bekommen, welche sich schneller/leichter auf Korrektheit überprüfen lässt wozu die folgenden Normalformen sehr hilfreich sind.

2 Vereinfachungen

In dem ersten Abteil möchten wir zunächst ein paar simple Vereinfachungen von Formeln wie zum Beispiel $\top \vee \varphi$ zu $\top$ vornehmen. Das machen/macht für uns die Funktionen/Funktion **psimplify1** und **psimplify**.

3 Negationsnormalform (NNF)

Definition:
Eine Formel φ ist in Negationsnormalform (NNF), falls sie die Symbole $\rightarrow$,$\leftrightarrow$ nicht enthält und $\neg$ nur vor atomaren Formeln steht.
In anderen Worten: Eine Formel ist in NNF, wenn sie lediglich aus Literalen mittels $\vee$ und $\wedge$ konstruiert wurde.

Definition Literal: Eine atomare Formel oder die negation einer atomaren Formel.

Nun möchten wir eine beliebige Formel in NNF überführen. Die Symbole $\rightarrow$ und $\leftrightarrow$ kriegen wir durch die Transformationen

$$(\varphi \leftrightarrow \psi) \text{ zu } (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$$
$$(\phi \rightarrow \psi) \text{ zu } (\neg\varphi \vee \psi)$$

und andere Formeln auf anderen wegen (Beispielhaft seien genannt:)

$$\neg(p \wedge q) \text{ zu } \neg p \vee \neg q$$
$$\neg(p \vee q) \text{ zu } \neg p \wedge \neg q$$

Schritt für Schritt in NNF überführt.

Dadurch erhalten wir die Funktion **nnf**.

ein Beispiel zeigt eine Problematik an diesem Verfahren auf:
Angenommen wir definieren fm nun auf folgende Weise:

```
let fm = <<(p <=> q) <=> ~(r ==> s)>>;;
```

Dann ist unsere Ausgabe wenn wir die Funktion nnf auf fm anwenden folgende:

```
<<(p /\ q \/ ~p /\ ~q) /\ r /\ ~s \/
  (p /\ ~q \/ ~p /\ q) /\ (~r \/ s)>>
```

WICHTIG:
Wir sehen jetzt, dass die Formel in NNF viel länger ist als unsere ursprüngliche Formel. Das liegt wenn wir in unsere Funktion nochmal reinschauen an den Formeln der Form $p \leftrightarrow q$ und $\neg(p \leftrightarrow q)$, denn dort verdoppeln sich p bzw. q jedes mal beim Übergang in NNF. Dieses verdoppeln kann im Worst Case exponentiell schnell geschehen weswegen wir eine weitere Funktion nämlich **nenf** haben, welche $\leftrightarrow$-Symbole anders behandelt (nämlich behält) und laut Harrison später noch nützlich wird.

4 Disjunktive und Konjunktive Normalform

Definition Eine Formel φ ist in *disjunktiver Normalform* (DNF), falls

$$\varphi = D_1 \vee ... \vee D_n$$

wobei wiederum jedes D_i von der Form

$$l_{i_1} \wedge ... \wedge l_{i_{m_i}}$$

für Literale l_{i_j} ist.
Anzumerken ist, dass jede Formel in DNF auch In NNF ist allerdings nicht zwangsweise andersherum. Gleiches gilt für die konjuktive Normalform, wobei eine Formel φ in konjunktiver Normalform ist, falls

$$\varphi = C_1 \wedge ... \wedge C_n$$

wobei jedes C_i von der Form

$$l_{i_1} \vee ... \vee l_{i_{m_i}}$$

für Literale l_{i_j} ist.
Wir wollen nun eine Formel in DNF bringen. (Danach können wir das Verfahren leicht abwandeln und es funktioniert dann auch für die KNF)

Im folgenden steht oft etwas davon ob eine Formel “allgemein wahr“ ist. Was gemeint ist, ist ob eine Formel unter jeder belegung der darin enthaltenen atomaren Formel mit true und false wahr ist.

4.1 Disjunktive Normalform über Wahrheitstabellen

Nehmen wir an eine Formel φ bestehe aus den Atomen $\{p_1, ..., p_n\}$ dann können wir φ auf eine sehr intuitive Art und Weise in DNF bringen:

Nehmen wir zunächst als Beispiel die Formel $\varphi = \neg(p_1 \vee \neg(\neg p_2 \wedge p_3)) \vee (p_3 \wedge \neg(\neg p_2 \vee p_1))$ Dann können wir für jede möglichen Werte von p_1 bis p_3 φ auswerten und in die Wahrheitstabelle eintragen, dann erhalten wir:

p_1	p_2	p_3	$\neg(p_1 \vee \neg(\neg p_2 \wedge p_3))$	$(p_3 \wedge \neg(\neg p_2 \vee p_1))$	φ
false	false	false	false	false	false
false	false	true	true	false	true
false	true	false	false	false	false
false	true	true	false	true	true
true	false	false	false	false	false
true	false	true	false	false	false
true	true	false	false	false	false
true	true	true	false	false	false

Nun sehen wir, dass φ wahr ist genau dann wenn Zeile 2 oder 4 eintreten. Wir nehmen jetzt also Zeile 2 und 4 und definieren:

$$l_{i1} := \begin{cases} p_i \text{ ,falls } p_i = true \text{ in Zeile 2} \\ \neg p_i \text{ sonst} \end{cases} \qquad l_{i2} := \begin{cases} p_i \text{ ,falls } p_i = true \text{ in Zeile 4} \\ \neg p_i \text{ sonst} \end{cases}$$

Allgemein gesprochen nehmen wir also nun die Zeilen in denen φ wahr ist und konstruieren die Literale analog zu der obigen Definition. Dann definieren wir für jede positive Zeile das dementsprechende D_j durch:

$$D_j := l_{1j} \wedge ... \wedge l_{nj}$$

dann kriege wir nun eine logisch äquivalente Formel zu φ indem wir alle D_j's durch disjunkte aneinanderheften und können dann wieder von der allgemeinen Beschreibung in unser Beispiel gehen:

$$\begin{aligned} & D_1 \vee ... \vee D_m \\ & \quad \rightarrow \; D_1 \vee D_2 \\ & \quad \rightarrow \; (\neg p_1 \wedge \neg p_2 \wedge p_3) \vee (\neg p_1 \wedge p_2 \wedge p_3) \end{aligned}$$

Um das im Code zu gewährleisten bauen wir uns zuerst zwei Funktionen, welche aus einer gegebenen Liste von Formeln eine neue Formel bauen einmal ausschließlich durch Konjunktionen (**list_conj**) und im anderen Fall durch Disjunktionen (**list_disj**).

Als nächstes brauchen wir um unsere Literale später wie oben passend zu wählen eine Funktion die eine Formelmenge unter einer gegebenen Variablenbelegung v (streng genommen keine Variablenbelegung deshalb von Harrison valuation gennant) passend zusammenbaut (**mk_lits**).

Nun brauchen wir noch eine Funktion die uns eine Liste aller derjenigen Valuations gibt, unter welchen unsere Formel φ erfüllt ist, damit wir diese nachher Nutzen können um die D_j's passend zu wählen und durch **list_disj** später wieder zusammenfügen können. (dies übernimmt **allsatvaluations**)

Nun können wir endlich die gewünschte Funktion zum erhalten der DNF implementieren (**dnf**):

1. Wir nehmen alle atomaren formeln von unserer Ausgangsformel (pvs)
2. Wir holen uns mit Hilfe von allsatvaluations alle valuations unter denen die Ausgangsformel wahr ist.
3. Wir nutzen alle valuations unter denen die Ausgangsformel wahr ist um die einzelnen Disjunkte zu bauen, welche wir dann durch **list_disj** wieder zusammenfügen.

WICHTIG:
Auch hier haben wir im Worst Case wieder ein exponentielles Wachstum der berechneten DNF der Formel φ in der Anzahl der atomaren Formeln in φ! (Also kein in der Praxis guter Algorithmus)

4.2 Disjunktive Normalform über Transformationen

Intuition: Die Formel ist nun in NNF und wird in DNF transformiert.

Ein anderer Weg um eine Formel in disjunktive Normalform zu bringen ist das nutzen des distributiv Gesetzes:

$$\begin{aligned} p \wedge (q \vee r) &\Leftrightarrow (p \wedge q) \vee (p \wedge r) \\ (p \vee q) \wedge r &\Leftrightarrow (p \wedge r) \vee (q \wedge r) \end{aligned}$$

Um eine Funktion zu implementieren die das gewährleistet bauen wir zunächste eine andere Funktion (**distrib**) in der wir davon ausgehen, dass dort die Formeln p,q und r bereits in DNF sind und verwenden dann nur noch die distributiv gesetze.

danach können wir nun die Funktion bauen, die uns tatsächlich zur disjunktiven Normalform führt indem wir den Fall der Oder Verknüpfung nun ebenfalls betrachten (**rawdnf**)

WICHTIG:
Nun haben wir allerdings das Problem, dass die ausgegebene Formel zwar in DNF aber recht unleserlich ist. Außerdem sind manche Disjunktionen eventuell komplett irrelevant je nach Ausgangsformel. (zum Beispiel wäre $p \wedge \neg p \wedge ....$ ein irrelevantes disjunkt, da es nie erfüllt werden kann)

4.3 Mengenbasierte Darstellung

Um weniger mit der Klammerung der Ausdrücke zu tun zu haben und leichter mit Vereinfachungen klar zu kommen kann man die DNF auch als Menge von Mengen verstehen. Bedeutet wir bilden eine Menge über die einzelnen disjunkte, welche wiederrum eine Menge über den in ihnen enthaltenen Literalen sind. Ein Beispiel hierfür ist $p \wedge q \vee \neg p \vee r$ in $\{\{p, q\}, \{\neg p, r\}\}$ zu übertragen. Dafür gibt es die Funktion **puredn**f die uns nun eine Liste von Listen ausgibt.

Nun wollen wir noch aus der DNF Disjunkte entfernen, welche ohnehin nicht erfüllbar sind wie zum Beispiel $p \wedge r \wedge \neg p$. Dafür schreiben wir eine Funktion, welche überprüfen soll ob wir in einer Liste komplementäre Literale haben (denn ein Disjunkt ist genau dann allgemein erfüllt, wenn komplementäre literale auftauchen). Dafür teilen wir die Literale in positive und negative Literale ein und schauen ob wir bei den positiven Literalen irgendwelche finden, welche negiert auch bei den negativen Literalen auftauchen. Diese Aufgabe erfüllt nun die Funktion trivial.

Um weiter zu vereinfach können wir Subsumption verwenden:
Bedeutet für zwei Mengen von Literalen mit $\{l'_1, ..., l'_m\} \subseteq \{l_1, ..., l_n\}$, dass jede Valuation, welche $D = l_1 \wedge ... \wedge l_n$ erfüllt automatisch auch $D' = l'_1 \wedge ... \wedge l'_m$ erfüllt. Also können wir $D \vee D'$ zu D' transformieren. Man sagt auch D' subsumiert D oder D wird von D' subsumiert. **simpdnf** ist jetzt die Funktion, welche diese Aufgabe für eine gegebene Formel die bereits in NNF ist gewährleistet.

4.4 Konjunktive Normalform

Für die KNF können wir analog eine Listen basierte Repräsentation nutzen. Hierbei sollten wir die De Morganschen Regeln beachten:

$$\neg p \Leftrightarrow \bigvee_{i=1}^{m} \bigwedge_{j=1}^{n} p_{ij}$$

genau dann wenn

$$p \Leftrightarrow \bigwedge_{i=1}^{m} \bigvee_{j=1}^{n} \neg p_{ij}$$

Also können wir die KNF nun über das bilden einer DNF erhalten:
Zuerst bilden wir die Negation der Startformel und bringen diese wieder in NNF.
Danach bilden wir von dem Ergebnis die DNF und negieren dort dann alle Literale.
(Funktion: **purecnf**)

Nun können wir auf analoge Weise auch hier wieder mit Subsumption arbeiten und dadurch die vereinfachte KNF erhalten. (Funktion: **simpcnf**)

4.5 Wichtige Beobachtungen

Die DNF bzw. die KNF bringen beide jeweils Vor- bzw. Nachteile mit sich. Der Vorteil der DNF ist, dass wir eine formel in DNF sehr schnell auf erfüllbarkeit testen können. In der KNF wiederrum können wir eine Formel sehr schnell auf ihren Wahrheitswert überprüfen.

Auch hier sei wieder erwähnt, dass dies in der Praxis keine guten Algorithmen sind aufgrund des exponentiellen Wachstum in der Anzahl der beinhalteten atomaren Formeln.

4.6 Aufgabe

Die Aufgabe ist es eine neue tautology Funktion zu schreiben. Dafür sollt ihr die KNF verwenden. Im Buch von Harrison wird dafür folgende Aussage verwendet:

Eine Formel in KNF ist genau dann allgemein wahr wenn jedes der Konjunkte allgemein wahr ist. Für ein Konjunkt $l_1 \vee ... \vee l_n$ gilt, dass es wahr ist genau dann wenn es Literale l_i und l_j gibt,so dass die Negation von l_i das allgemeine logische Äquivalent von l_j ist.
