Automatisches Beweisen—Vertiefung
First Order Logic

Christoph Zengler

Arbeitsbereich Symbolisches Rechnen
Prof. Dr. Wolfgang Küchlin
Universität Tübingen

18. Oktober 2011

Logischer Formalismus

Syntax

Wie werden Formeln gebildet?

- klassisch-mathematisch: bestimmte ausgezeichnete Zeichenreihen
- informatisch: Sprache einer Grammatik

Semantik

Was ist die Bedeutung einer Formel?

- Allgemein: Abbildung in einen (bekannten) Semantik-Bereich

Kalkül

Wie kann der Wahrheitswert einer Formel bestimmt werden?

- Inferenzregeln, Algorithmen

Aussagenlogik vs. Logik erster Stufe

Aussagenlogik

- Nur atomare Aussagen, die wahr oder falsch sind, z.B.
  - *“5 ist eine Primzahl”*
  - *“7 ist gerade”*
  - *“Thomas ist ein Student”*
- Variablen können nicht quantifiziert werden

Logik erster Stufe

- Atome sind Prädikate, die aus Termen und Variablen über einem Universum bestehen, z.B.
  - $x < y$
  - $\texttt{istStudent}(x)$
  - $f(x, y) \geqslant g(x) + g(y)$
- Variablen können mit $\exists$ und $\forall$ quantifiziert werden

Syntax der First Order Logic (FOL)

EBNF-Grammatik für Formeln erster Stufe

Term	$=$	*Termvariable*	$\in \mathcal{V}_T$
	$\mid$	*Funktionssymbol*({Term})	Funktionsapplikation

Formel	$=$	$\top \mid \bot$	Konstanten
	$\mid$	$\neg$ Formel	Negation
	$\mid$	*Prädikatssymbol*({Term})	Prädikatapplikation
	$\mid$	Formel $\wedge$ Formel	Konjunktion
	$\mid$	Formel $\vee$ Formel	Disjunktion
	$\mid$	Formel $\rightarrow$ Formel	Implikation
	$\mid$	Formel $\leftrightarrow$ Formel	Äquivalenz
	$\mid$	$\forall$ Termvariable [Formel]	Allquantor
	$\mid$	$\exists$ Termvariable [Formel]	Existenzquantor
	$\mid$	(Formel)	Klammerung

Anmerkungen zur Syntax

- Unter Atomaren Formeln fassen wir Prädikate und Konstanten zusammen
- Jedes Funktionssymbol und jedes Prädikatssymbol hat eine Stelligkeit (Arität), die angibt wie viele Terme als Parameter übergeben werden
  - Notation $f^{(2)}$: Funktionssymbol f hat Stelligkeit 2
- Funktionen mit Stelligkeit 0 nennen wir auch Konstanten, Prädikate mit Stelligkeit 0 nennen wir Boolesche Konstanten
- Formeln repräsentieren Wahrheitswerte, Terme repräsentieren Objekte (z.B. Zahlen, Studenten, . . .)
- Relationen werden auch Prädikate genannt
- Logik erster Stufe wird auch Prädikatenlogik genannt
- In FOL dürfen wir nur über Termvariablen quantifizieren, will man z.B. über Variablen, Funktionen oder Prädikate quantifizieren, muss man eine Logik höherer Ordnung (HOL) benutzen.

First Order Logic—Beispiele

Beispiel (Syntax der FOL)

Sprache: $(+^{(2)}, -^{(1)}, 0^{(0)}, \leqslant^{(2)}, =^{(2)})$

Wohlgeformte Terme:

- $x + 0$, $-x$, -0, $x + y$

Wohlgeformte Atomare Formeln

- $x + 0 \leqslant -x$
- $x + 0 = 0$
- $x + -y \leqslant -z$

Wohlgeformte Formeln

- $x + 0 \leqslant -x \wedge x + 0 = 0$
- $\exists z[x + -y \leqslant -z]$

Quantoren

- Quantoren $\exists x$ und $\forall x$ binden die Variable x (lokale Definition)
- Eine Variable kann gleichzeitig gebunden und frei in einer Formel vorkommen

Definition (Freie und gebundene Variablen)

- Menge der freien Variablen einer Formel φ: $\mathtt{free}(\varphi)$
- Menge der gebundenen Variablen einer Formel φ: $\mathtt{bound}(\varphi)$

Beispiel (Freie und gebundene Variablen)

$$\varphi = \exists a[a = 12 \wedge b > a \wedge \forall u \exists b[b + u = 24 \wedge x = y]]$$

- $\mathtt{free}(\varphi) = \{b, x, y\}$
- $\mathtt{bound}(\varphi) = \{a, b, u\}$

- Wir erlauben die abkürzende Schreibweise $Q(x_1, \ldots, x_n)[\varphi]$ für $Qx_1[\ldots[Qx_n[\varphi]]]$ mit $Q \in \{\exists, \forall\}$

Semantik der FOL

1. Belegung $\beta_D : \mathcal{V}_T \to D$ der Termvariablen
2. Interpretation $\mathcal{M}$ bestehend aus 3 Teilen:

Definition (Das Universum D)

nicht-leere Menge D (domain) der Individuen, über die man spricht. D.h. alle Terme evaluieren zu einem Wert in D.

Definition (Interpretation der Funktionssymbole)

Jedem n-stelligen Funktionssymbol f wird eine Funktion $f_{\mathcal{M}} : D^n \to D$ zugeordnet

Definition (Interpretation der Prädikatssymbole)

Jedem n-stelligen Prädikatssymbol P wird eine Boolesche Funktion $P_{\mathcal{M}} : D^n \to \{\texttt{true}, \texttt{false}\}$ zugeordnet, d.h. $P_{\mathcal{M}} \subseteq D^n$

Evaluation von Termen

Algorithmus: $\texttt{termval}(M, \beta_D, t)$

Eingabe: Interpretation M, Belegung β_D, Term t
Ausgabe: Evaluation von t unter D und β_D

$\texttt{termval}(M, \beta_D, t) = t \; \textit{match}$

$\quad x \in \mathcal{V}_T \rightsquigarrow \beta_D(x)$

$\quad \mid f(t_1, \ldots, t_n) \rightsquigarrow f_M(\texttt{termval}(M, \beta_D, t_1), \ldots, \texttt{termval}(M, \beta_D, t_n))$

Beispiel (Evaluation von Termen)

- Funktionssymbole: $\diamond^{(0)}, \ominus^{(1)}, \oplus^{(2)}$
- Interpretation M mit Universum $\mathbb{Z}$ und Interpretationen
  - $\diamond_M = 0$
  - $\ominus_M(x) = -x$
  - $\oplus_M(x, y) = x + y$
- $\beta_D = \{x \mapsto 2, y \mapsto -4\}$

$\texttt{termval}(\oplus(x, \ominus(\oplus(y, \diamond)))) = 6 \in \mathbb{Z}$

Evaluation von FOL Formeln

Algorithmus: $\texttt{holds}(M, \beta_D, \psi)$

Eingabe: Interpretation M, Belegung β_D der Termvariablen, Formel ψ
Ausgabe: Evaluation von ψ unter M und β_D

$\texttt{holds}(M, \beta_D, \psi) = \psi$ *match*

$\top \rightsquigarrow$ `true`

$\mid \bot \rightsquigarrow$ `false`

$\mid P(t_1, \ldots, t_n) \rightsquigarrow P_M(\texttt{termval}(M, \beta_D, t_1), \ldots, \texttt{termval}(M, \beta_D, t_n))$

$\mid \neg\varphi \rightsquigarrow$ *if* $\texttt{holds}(M, \beta_D, \varphi)$ *then* `false` *else* `true`

$\mid \varphi_1 \wedge \varphi_2 \rightsquigarrow$ *if* $\texttt{holds}(M, \beta_D, \varphi_1)$ *and* $\texttt{holds}(M, \beta_D, \varphi_2)$ *then* `true` *else* `false`

$\mid \varphi_1 \vee \varphi_2 \rightsquigarrow \texttt{holds}(M, \beta_D, \neg(\neg\varphi_1 \wedge \neg\varphi_2))$

$\mid \varphi_1 \rightarrow \varphi_2 \rightsquigarrow \texttt{holds}(M, \beta_D, \neg\varphi_1 \vee \varphi_2)$

$\mid \varphi_1 \leftrightarrow \varphi_2 \rightsquigarrow \texttt{holds}(M, \beta_D, \varphi_1 \rightarrow \varphi_2 \wedge \varphi_2 \rightarrow \varphi_1)$

$\mid \forall x[\varphi] \rightsquigarrow$ *for all* $a \in D$: $\texttt{holds}(M, \beta_D \cup \{x \mapsto a\}, \varphi)$

$\mid \exists x[\varphi] \rightsquigarrow$ *exists* $a \in D$: $\texttt{holds}(M, \beta_D \cup \{x \mapsto a\}, \varphi)$

Evaluation von FOL Formeln—Beispiel

Beispiel (Evaluation von FOL Formeln)

- Funktionssymbole: $\diamond^{(0)}, \ominus^{(1)}, \oplus^{(2)}$
- Prädikatsssymbole: $\sqsubseteq^{(2)}, \doteq^{(2)}$
- Interpretation $\mathcal{M}$ mit Universum $\mathbb{Z}$ und Interpretationen
  - $\diamond_{\mathcal{M}} = 0$
  - $\ominus_{\mathcal{M}}(x) = -x$
  - $\oplus_{\mathcal{M}}(x, y) = x + y$
  - $\sqsubseteq_{\mathcal{M}}(x, y) = x \leqslant y$
  - $\doteq_{\mathcal{M}}(x, y) = x \neq y$
- $\beta_D = \{x \mapsto 2, y \mapsto -4\}$
- $\psi = \sqsubseteq(\oplus(x, \ominus(\oplus(y, \diamond))), \ominus(x)) \vee \neg(\doteq(\oplus(x, y), \diamond))$

$$
\begin{aligned}
\texttt{holds}(\mathcal{M}, \beta_D, \psi) &= (x + -(y + 0) \leqslant -x) \vee \neg(x + y \neq 0) \\
&= (2 + -(-4 + 0) \leqslant -2) \vee \neg(2 + -4 \neq 0) \\
&= 6 \leqslant -2 \vee -2 = 0 \\
&= \texttt{false}
\end{aligned}
$$

Freie Variablen

Theorem (Formeln unter gleicher Belegung der freien Variablen)

Gilt für eine Formel φ

- *für alle* $x \in \texttt{free}(\varphi)$, *dass* $\beta_D(x) = \beta'_D(x)$

so gilt auch

- $\texttt{holds}(\mathcal{M}, \beta_D, \varphi) = \texttt{holds}(\mathcal{M}, \beta'_D, \varphi)$ *für beliebige Interpretationen* $\mathcal{M}$.

- Ein Term oder eine Formel ohne Variablen heißt variablenfrei oder ground.
- Eine Formel ohne freie Variablen heißt Satz.

Theorem (Sätze unter beliebigen Belegungen)

Ist φ *ein Satz, so gilt für beliebige Belegungen* β_D *und* β'_D, *dass* $\texttt{holds}(\mathcal{M}, \beta_D, \varphi) = \texttt{holds}(\mathcal{M}, \beta'_D, \varphi)$.

Gültigkeit

Definition (Gültigkeit)

Eine FOL Formel φ heißt gültig oder valide, wenn für alle möglichen Interpretationen $\mathcal{M}$ und Belegungen β_D gilt, dass $\texttt{holds}(\mathcal{M}, \beta_D, \varphi) = \texttt{true}$.

Beispiel (Gültige Formel)

$$\forall x[P(x)] \rightarrow P(a)$$

(gilt ein Prädikat P für alle Objekte, so gilt es auch für ein spezielles Objekt a)

Vorsicht!

Der Quantor $\forall x$ und sein Scope sind wichtig. Weder

- $P(x) \rightarrow P(a)$ noch
- $\forall x[P(x) \rightarrow P(a)]$

sind gültig.

Erfüllbarkeit

Definition (Erfüllbarkeit)

Eine Interpretation $\mathcal{M}$ erfüllt eine FOL Formel φ (φ gilt in $\mathcal{M}$), wenn für *alle* Belegungen β_D gilt, dass $\texttt{holds}(\mathcal{M}, \beta_D, \varphi) = \texttt{true}$.

- $\mathcal{M}$ erfüllt eine Menge von FOL Formeln Γ (Γ gilt in $\mathcal{M}$), wenn $\mathcal{M}$ jede Formel $\varphi \in \Gamma$ erfüllt
- Eine FOL Formel φ ist erfüllbar, wenn eine Interpretation $\mathcal{M}$ *existiert*, die sie erfüllt.

Theorem (Gültigkeit von Sätzen)

Ein Satz φ ist gültig, genau dann wenn $\neg\varphi$ unerfüllbar ist.

⚠ Vorsicht!

Obiger Satz gilt nicht für Formeln mit freien Variablen:

$P(x) \vee \neg P(y)$ ist nicht gültig, $\neg P(x) \wedge P(y)$ ist jedoch unerfüllbar.

Modelle

Definition (Modell)

Eine Interpretation $\mathcal{M}$, die eine Menge Γ von Formeln erfüllt, heißt Modell von Γ.

Notationen:

- $\Gamma \models \varphi$: φ gilt in allen Modellen von Γ
- Wir schreiben $\models \varphi$ statt $\emptyset \models \varphi$ (gilt in allen Modellen)
- $\models_{\mathcal{M}} \varphi$: $\mathcal{M}$ erfüllt φ

Folgerungen:

- Γ ist unerfüllbar, gdw. $\Gamma \models \bot$

⚠ Vorsicht!

Im Gegensatz zur Aussagenlogik, gilt in FOL nicht, dass $\{\varphi_1, \ldots, \varphi_n\} \models \varphi$ äquivalent ist zu $\models \varphi_1 \wedge \cdots \wedge \varphi_n \rightarrow \varphi$

Abschlüsse

Oft is es angenehmer mit Sätzen anstelle von beliebigen Formeln zu arbeiten. Wenn z.B. alle Formeln φ_i Sätze sind, so gilt auch in FOL

$$\{\varphi_1, \ldots, \varphi_n\} \models \varphi \text{ ist äquivalent zu } \models \varphi_1 \wedge \cdots \wedge \varphi_n \rightarrow \varphi.$$

Definition (Universeller Abschluss)

Allquantifiziere alle freien Variablen einer Formel φ.

- Notation: $\underline{\forall}\varphi$

- φ ist gültig, gdw. $\underline{\forall}\varphi$ gültig ist.

Definition (Existentieller Abschluss)

Existenzquantifiziere alle freien Variablen einer Formel φ.

- Notation: $\underline{\exists}\varphi$

Substitution in Termen

Konvention: Substitution σ bildet immer alle freien Variablen einer Formel ab. Variablen, die nicht substitutiert werden, werden auf sich selber abgebildet.

Algorithmus: $\texttt{tsubst}(\sigma, t)$

Eingabe: Abbildung σ von Variablen $\in \mathcal{V}_T$ zu Termen, Term t
Ausgabe: Term $t\sigma$ (Substitution auf t ausgeführt)

$\texttt{tsubst}(\sigma, t) = t$ *match*

$x \in \mathcal{V}_T \rightsquigarrow \sigma(x)$

$\mid f(t_1, \ldots, t_n) \rightsquigarrow f(\texttt{tsubst}(\sigma, t_1), \ldots, \texttt{tsubst}(\sigma, t_n))$

Beispiel (Substitution in Termen)

- $t = (x + 24) \cdot (y - a)$
- $\sigma = \{x \mapsto 12, y \mapsto x + z\}$

$$\texttt{tsubst}(\sigma, t) = (12 + 24) \cdot ((x + z) - a)$$

Substitution in Formeln

⚠ Nicht so einfach wie gedacht!

- Gebundene Variablen dürfen nicht substitutiert werden
- Substitution kann Variablen einführen, die bereits gebunden sind

Beispiel (Problemfälle)

- Substitution von x in $\forall x[x = x]$ darf keine Auswirkung haben
- Substitution $\{y \mapsto x\}$ in $\exists x[x + 1 = y]$ würde aus freier Variable y eine gebundene machen

Lösungsidee

- Gebundene Variablen sind nur Platzhalter

$\Rightarrow$ können umbenannt werden (Alpha Konversion)

Substitution in Formeln—Vorgehen

Algorithmus: $\mathtt{subst}(\sigma, \psi)$

Eingabe: Substitution σ, Formel ψ
Ausgabe: $\psi\sigma$

$\mathtt{subst}(\sigma, \psi) = \psi$ *match*

$\top \rightsquigarrow \top$

$\mid \bot \rightsquigarrow \bot$

$\mid P(t_1, \ldots, t_n) \rightsquigarrow P(\mathtt{tsubst}(\sigma, t_1), \ldots, \mathtt{tsubst}(\sigma, t_n))$

$\mid \neg\varphi \rightsquigarrow \neg\mathtt{subst}(\sigma, \varphi)$

$\mid \varphi_1 \wedge \varphi_2 \rightsquigarrow \mathtt{subst}(\sigma, \varphi_1) \wedge \mathtt{subst}(\sigma, \varphi_2)$

$\mid \varphi_1 \vee \varphi_2 \rightsquigarrow \mathtt{subst}(\sigma, \varphi_1) \vee \mathtt{subst}(\sigma, \varphi_2)$

$\mid \varphi_1 \to \varphi_2 \rightsquigarrow \mathtt{subst}(\sigma, \varphi_1) \to \mathtt{subst}(\sigma, \varphi_2)$

$\mid \varphi_1 \leftrightarrow \varphi_2 \rightsquigarrow \mathtt{subst}(\sigma, \varphi_1) \leftrightarrow \mathtt{subst}(\sigma, \varphi_2)$

$\mid \forall x[\varphi] \rightsquigarrow \mathtt{substq}(\sigma, \forall, x, \varphi)$

$\mid \exists x[\varphi] \rightsquigarrow \mathtt{substq}(\sigma, \exists, x, \varphi)$

$\mathtt{substq}(\sigma, q, v, \varphi) =$

$x = $ *if exists* $y \neq v \in \mathtt{free}(\varphi)$ *with* $v \in \mathtt{free}(\sigma(y))$ *then* v' *else* v

return $qx[\mathtt{subst}(\sigma \cup \{v \mapsto x\}, \varphi)]$

Substitution in Formeln—Eigenschaften

⚠ Simultane Substitution

Substitution ist simultan, d.h. alle Substitutionen werden gleichzeitig ausgeführt.

Lemma (Substitutions Lemma)

Für eine beliebige Formel φ, Substitution σ, Interpretation $\mathcal{M}$ und Belegung β_D gilt:

$$\texttt{holds}(\mathcal{M}, \beta_D, \texttt{subst}(\sigma, \varphi)) = \texttt{holds}(\mathcal{M}, \beta_D \circ \sigma, \varphi)$$

Corollary

Wenn eine Formel φ gültig ist, so auch $\varphi\sigma$ für beliebige Substitutionen σ.

Notationen:

- Wir sagen statt Substitution auch oft Instanziierung
- Schreibweise: $\varphi\sigma$ statt $\texttt{subst}(\sigma, \varphi)$

Substitution in Formeln—Demo

Demo

```
import org.warthog.formulas._
import org.warthog.formulas.types._

val equ = Variable("eq", CompositeType(BooleanType, IndividualType,
IndividualType))
val add = Variable("add", CompositeType(IndividualType, IndividualType,
IndividualType))
val one = Variable("1", CompositeType(IndividualType))
val x = Variable("x", IndividualType)
val y = Variable("y", IndividualType)

val phi1 = ForAll(x,Application(equ,x,x))
val phi2 = Exists(x,Application(equ, Application(add,x,Application(one)),y))

phi1.substitute(x,y)
... = ∀(x)[eq(x,x)]

phi2.substitute(y,x)
... = ∃(x0)[eq(add(x0,1()),x)]
```

Negationsnormalform

Definition (NNF)

Eine FOL Formel φ ist in NNF, wenn

1. nur die Booleschen Operatoren $\neg, \wedge$ und $\vee$ in φ vorkommen
2. $\neg$ nur vor Atomaren Formeln steht

Beispiel (Negationsnormalform)

- $\neg(P(x) \vee (P(y) \wedge \neg R(z)))$ nicht in NNF
- $\neg P(x) \wedge (\neg P(y) \vee R(z))$ ist in NNF

Idee!

Benutze Dualität der Quantoren:

$$\forall x[\varphi] \equiv \neg\exists x[\neg\varphi]$$

$$\exists x[\varphi] \equiv \neg\forall x[\neg\varphi]$$

Negationsnormalform—Algorithmus

Algorithmus: $\mathtt{nnf}(\psi)$

Eingabe: FOL Formel ψ
Ausgabe: NNF von ψ

$\mathtt{nnf}(\psi) = \psi$ *match*

alle Regeln von nnf *Algorithmus der Aussagenlogik +*

- $\forall x[\varphi] \rightsquigarrow \forall x[\mathtt{nnf}(\varphi)]$
- $\exists x[\varphi] \rightsquigarrow \exists x[\mathtt{nnf}(\varphi)]$
- $\neg\forall x[\varphi] \rightsquigarrow \exists x[\mathtt{nnf}(\neg\varphi)]$
- $\neg\exists x[\varphi] \rightsquigarrow \forall x[\mathtt{nnf}(\neg\varphi)]$

Pränexnormalform (PNF)

Definition (PNF)

Eine FOL Formel φ in NNF ist in PNF, wenn

- sie von der Form $Q_1 x_1\ Q_2 x_2 \ldots Q_n x_n \psi$ ist, mit
- $Q_i \in \{\exists, \forall\}$ und
- ψ quantorenfrei

d.h. alle Quantoren kommen nur ganz außen vor

Notation:

- der quantorenfreie Anteil ψ einer PNF Formel φ heißt auch Matrix. $\psi = \mathrm{mat}(\varphi)$.

Beispiel (PNF)

- $\exists x[P(x)] \rightarrow \exists y[P(y) \wedge \forall z[P(z)]]$ nicht in PNF
- $\forall x \exists y \forall z[P(x) \wedge P(y) \vee P(z)]$ ist in PNF

Pränexnormalform—Algorithmus

Implementierung sehr technisch, hier nur die Skizze

1. Bringe Formel zuerst in NNF
2. Quantoren werden sukzessive nach außen gezogen
   - Umbenennungen können nötig sein:
     - $P(x) \wedge \exists x[R(x)] \not\equiv \exists x[P(x) \wedge R(x)]$
     - $P(x) \wedge \exists x[R(x)] \equiv \exists x'[P(x) \wedge R(x')]$
   - Quantoren können in speziellen Fällen reduziert werden:
     - $\forall x[\varphi_1] \wedge \forall y[\varphi_2] \equiv \forall z[\varphi_1[x/z] \wedge \varphi_2[y/z]]$
     - $\exists x[\varphi_1] \vee \exists y[\varphi_2] \equiv \exists z[\varphi_1[x/z] \vee \varphi_2[y/z]]$

Übung

Beweisen Sie die folgenden Äquivalenzen:

- $\forall x[\varphi_1] \wedge \forall y[\varphi_2] \equiv \forall z[\varphi_1[x/z] \wedge \varphi_2[y/z]]$
- $\exists x[\varphi_1] \vee \exists y[\varphi_2] \equiv \exists z[\varphi_1[x/z] \vee \varphi_2[y/z]]$

Pränexnormalform—Eigenschaften & Demo

- PNF Konversion erhält freie Variablen und deren Namen
- Bei einer Formel φ in PNF ist die Matrix $\mathrm{mat}(\varphi)$ im Scope eines jeden Quantors

$\Rightarrow$ Keine Variable kann gleichzeitig frei und gebunden vorkommen: $\mathtt{bound}(\varphi) \cap \mathtt{free}(\varphi) = \emptyset$

Demo

```
import org.warthog.formulas._
import org.warthog.formulas.types._
import collection.immutable.SortedSet

val harP = Variable("P", CompositeType(BooleanType, IndividualType))
val harR = Variable("R", CompositeType(BooleanType, IndividualType))
val x = Variable("x", IndividualType)
val y = Variable("y", IndividualType)

val phi = Application(harP, x) && Exists(x,Application(harR,x))

phi.pnf
... = ∃(x0)[(P(x) ∧ R(x0))]
```

Die Unentscheidbarkeit der FOL

- **Aussagenlogik:** Entscheidung über Wahrheitswert einer Formel mit Hilfe von Wahrheitstafeln $\Rightarrow$ **entscheidbar**
- **First Order Logic:** Im Allgemeinen unmöglich! Es gibt Formeln, die nur unendliche Modelle besitzen.

Beispiel (Unentscheidbarkeit von FOL)

$\varphi = \forall(u, v, w)[(P(u, v) \wedge P(v, w)) \rightarrow P(u, w)] \wedge \forall x[\neg P(x, x)] \wedge \forall y[P(y, f(y))]$

Unendliches Modell mit Universum $\mathbb{N}$

- $f_{\mathcal{M}} = \mathbb{N} \rightarrow \mathbb{N} : n \mapsto n + 1$
- $P_{\mathcal{M}} = \{(x, y) \in \mathbb{N}^2 \mid x < y\}$
- β_D beliebig, da alle Variablen gebunden

Endliches Modell ist nicht möglich!

Theorem (Satz von Church)

Es gibt keinen Algorithmus, der für eine beliebige gegebene FOL Formel φ in endlich vielen Schritten entscheidet, ob φ erfüllbar ist, oder nicht.

... aber nicht verzagen!

Wir werden im Folgenden sehen, dass die FOL immerhin semi-entscheidbar ist.

Was heißt das?

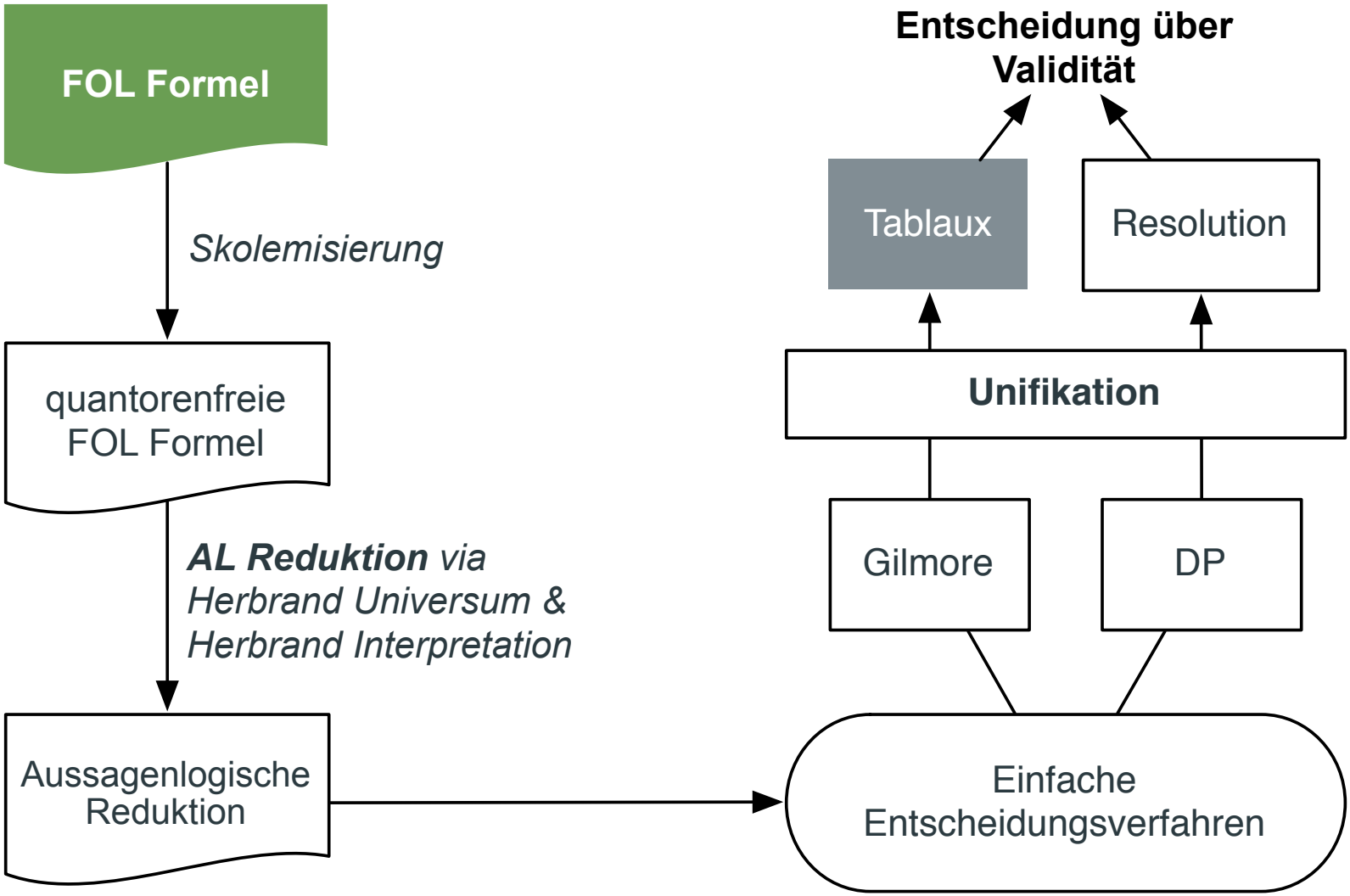
- Für eine unerfüllbare Formel können wir ein Verfahren angeben, das die Unerfüllbarkeit in endlicher Zeit nachweist.
- Problem: Wir können keine obere Grenze für die Laufzeit dieses Verfahrens angeben.

$\Rightarrow$ Wir wissen nicht, ob das Verfahren noch arbeitet, oder ob die Formel erfüllbar ist.

i Was können wir semi-entscheiden?

- Ist eine Formel eine Kontradiktion oder eine Tautologie (via Negation)
- Allgemeine Erfüllbarkeit kann nicht entschieden werden!

Der weitere Plan



Skolemisierung—Motivation

- PNF trennt Quantoren von der Matrix
- Immer noch beliebige Verschachtelungen von Quantoren möglich
- Skolemisierung: Eliminieren der Existenzquantoren

Hintergrund

Mathematisch äquivalent:

1. für alle $x \in D$ gibt es ein $y \in D$, so dass $P(x, y)$ gilt
2. es existiert ein $f : D \to D$, so dass für alle $x \in D$ gilt $P(x, f(x))$

Problem!

Formalisierung obiger Aussagen ist nicht mehr FOL:

$$\forall x \exists y [P(x, y)] \equiv \exists f \forall x [P(x, f(x))]$$

Ansonsten könnte man damit Existenz- und Allquantoren trennen:

$$\begin{aligned} \forall x \exists y \forall u \exists v [P(u, v, x, y)] &\equiv \exists f \forall (x, u) \exists v [P(u, v, x, f(x))] \\ &\equiv \exists (f, g) \forall (x, u) [P(u, g(x, u), x, f(x))] \end{aligned}$$

Skolemisierung—Beispiel

- Existenzielle Quantifizierung über Funktionen ist bereits implizit im Begriff der Erfüllbarkeit enthalten
  - *Eine Formel ist erfüllbar, wenn eine Interpretation existiert, die sie erfüllt*

$\Rightarrow$ Entstehende Formel ist nicht mehr logisch äquivalent, jedoch noch erfüllbarkeitsäquivalent.

Beispiel (Skolemisierung)

$$\forall x \exists y \forall u \exists v [P(u, v, x, y)]$$

wird zu

$$\forall (x, u) [P(u, g(x, u), x, f(x))]$$

und wegen implizter Allquantifizierung der freien Variablen zu

$$P(u, g(x, u), x, f(x))$$

- f und g heißen Skolemfunktionen und dürfen in der bisherigen Formel nicht vorkommen

Skolemisierung—Korrektheit

Definition (Menge der Funktionen)

Die Menge $\texttt{funcs}(\varphi)$ enthält alle Funktionssymbole der Formel φ.

Theorem (Korrektheit der Skolemisierung)

Ist φ eine Formel, mit

- $f \notin \texttt{funcs}(\varphi)$ *und*
- $\texttt{free}(\exists y[\varphi]) = \{x_1, \ldots, x_n\}$

dann gibt es zu einer beliebigen Interpretation $\mathcal{M}$ eine weitere Interpretation $\mathcal{M}'$, die sich nur in der Interpretation von f unterscheidet, so dass für alle Belegungen β_D gilt:

$$\texttt{holds}(\mathcal{M}, \beta_D, \exists y[\varphi]) = \texttt{holds}(\mathcal{M}', \beta_D, \varphi[y/f(x_1, \ldots, x_n)])$$

und ebenso

$$\texttt{holds}(\mathcal{M}, \beta_D, \exists y[\varphi]) = \texttt{holds}(\mathcal{M}', \beta_D, \exists y[\varphi]) \text{ , } \textit{da } f \notin \texttt{funcs}(\varphi).$$

Skolemisierung—Vorgehen

- $\exists x$ Quantoren werden sukzessive eliminiert
- Verschiedene Strategien denkbar:
  - Von außen nach innen oder von innen nach außen eliminieren
  - Formel zuerst in Normalformen bringen?

Normalformen

- NNF ist ratsam, da man sonst möglicherweise überflüssige Skolemisierungen vollzieht: $\neg\exists x[x = 4] \equiv \forall x[x \neq 4]$
- PNF überflüssig, da sie möglicherweise mehr freie Variablen in den Scope eines existentiellen Quantors einführt

Skolemisierungs Reihenfolge

- Von außen nach innen eliminieren reduziert ebenfalls Anzahl freier Variablen:

$$\exists x \exists y[x \cdot y = 1] \rightsquigarrow \exists y[sk_0 \cdot y = 1] \rightsquigarrow sk_0 \cdot sk_1 = 1$$

versus

$$\exists x \exists y[x \cdot y = 1] \rightsquigarrow \exists x[x \cdot sk_0(x) = 1] \rightsquigarrow sk_1 \cdot sk_0(sk_1) = 1$$

Skolemisierung—Implementierung & Demo

Algorithmus: skolemize(φ)

Eingabe: Formel φ
Ausgabe: Matrix der Skolemisierung von φ

1. Berechne $\varphi' = \mathtt{nnf}(\varphi)$
2. Skolemisiere in φ' Existenzquantoren von außen nach innen
3. Gib Matrix des Resultats zurück

Demo

```
phi1: ... = ∃(y)[(<(x,y) → ∀(u)[∃(v)[<(*(x,u),*(y,v))]])]
phi2: ... = ∀(x)[(P(x) → (∃(x,y)[Q(y)] ∨¬∃(z)[(P(z) ∧ Q(z))]))]

phi1.transform(FOLSkolemizationStrategy)
... = ¬<(x,sk0(x)) ∨ <(*(x,u0),*(sk0(x),sk1(u0,x)))

phi2.transform(FOLSkolemizationStrategy)
... = ¬P(x) ∨ Q(sk0()) ∨¬P(z0) ∨¬Q(z0)
```

Klauselnormalform

- Äquivalent zur CNF in der Aussagenlogik

Definition (Klauselnormalform)

Eine Formel ist in Klauselnormalform, wenn sie skolemisiert und in CNF ist.

Darstellung häufig als Klauselmenge und Literalmenge

Beispiel (Klauselnormalform)

$$(\forall x \exists y \forall u \exists v [P(u, v, x, y)] \vee R(z)) \wedge Q(z, w)$$

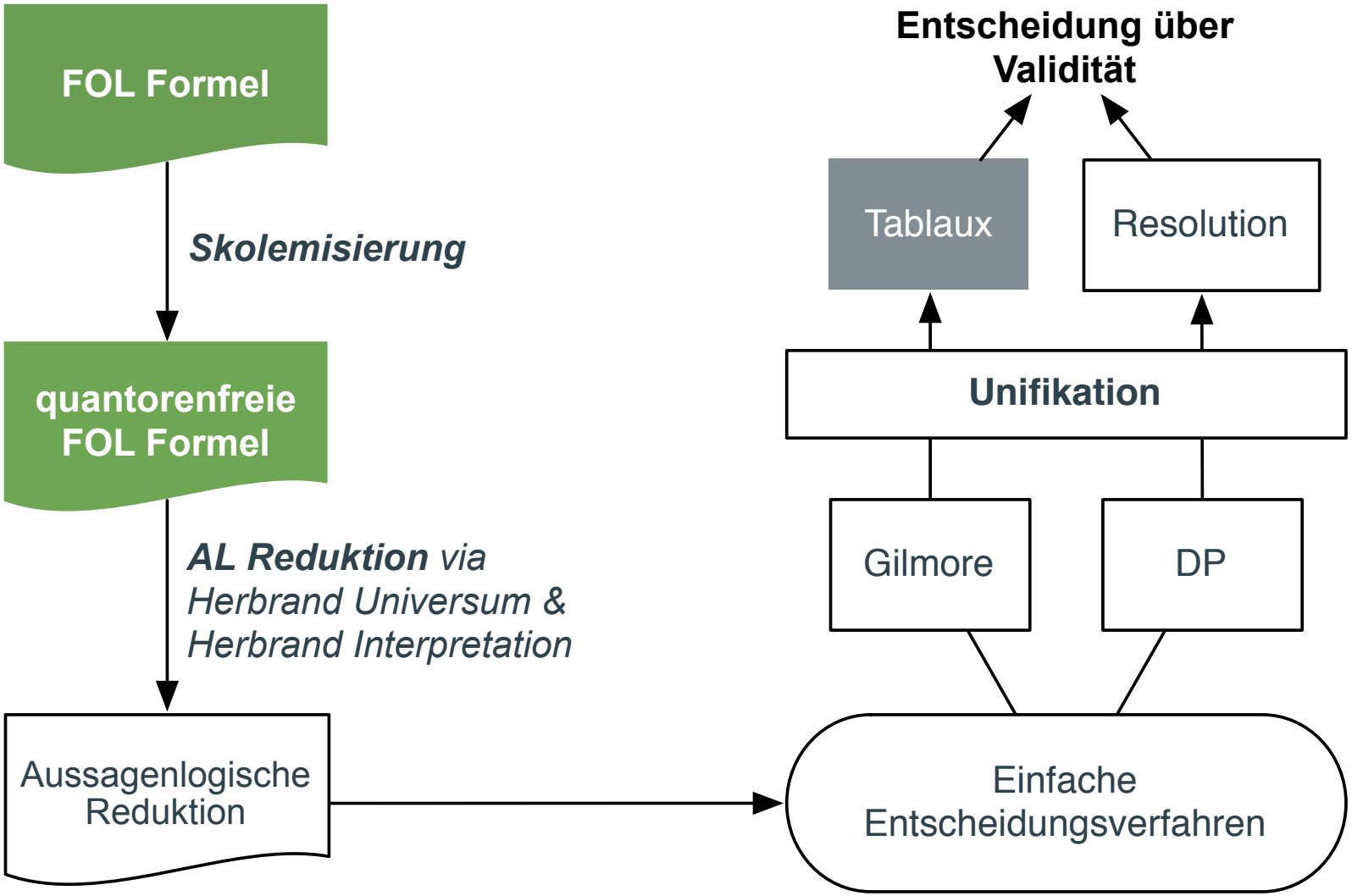
wird skolemisiert zu

$$(P(u, g(x, u), x, f(x)) \vee R(z)) \wedge Q(z, w)$$

und damit in Klauselnormalform zu

$$\{\{P(u, g(x, u), x, f(x)), R(z)\},\ \{Q(z, w)\}.$$

Wo stehen wir?



Kanonische Modelle

Quantorenfreie Formeln in FOL können auch aussagenlogisch betrachtet werden:

- Anstelle von aussagenlogischen Variablen haben wir Prädikatssymbole, die auf Terme angewandt werden
- Eine gegebene Formel hat immer eine endliche Anzahl an Variablen, Funktions- und Prädikatssymbolen
- Beweis der Kompaktheit kann angewandt werden

Idee!

FOL Formel:

$$P(a) \wedge R(x, y) \vee \neg R(x, f(x)) \wedge P(a)$$

AL Äquivalenz:

$$A \wedge B \vee \neg C \wedge A$$

? Interessante Frage!

Kann man die FOL-Validität zurückzuführen auf die aussagenlogische Validität?

Aussagenlogische Gültigkeit

- $\beta_{\mathcal{A}}$: Abbildung von atomaren Formeln auf Wahrheitswerte

Algorithmus: $\texttt{pholds}(\beta_{\mathcal{A}}, \varphi)$

Eingabe: $\beta_{\mathcal{A}}$ wie oben definiert, quantorenfreie FOL Formel φ
Ausgabe: `true`, wenn φ in aussagenlogischer Betrachtung gilt, sonst `false`

$\texttt{pholds}(\beta_{\mathcal{A}}, \varphi) = \varphi$ *match*

| φ ist atomare Formel $\rightsquigarrow$ $\beta_{\mathcal{A}}(\varphi)$

| ... $\rightsquigarrow$ alle anderen Fälle wie bei $\texttt{eval}(\beta, \varphi)$

Beispiel (Aussagenlogische Gültigkeit)

- $\varphi = P(a) \wedge R(x, y) \vee \neg R(x, f(x)) \wedge P(a)$
- $\beta_{\mathcal{A}} = \{P(a) \mapsto \texttt{true}, R(x, y) \mapsto \texttt{false}, R(x, f(x)) \mapsto \texttt{false}\}$

$$\texttt{pholds}(\beta_{\mathcal{A}}, \varphi) = \texttt{true} \wedge \texttt{false} \vee \texttt{true} \wedge \texttt{true} = \texttt{true}$$

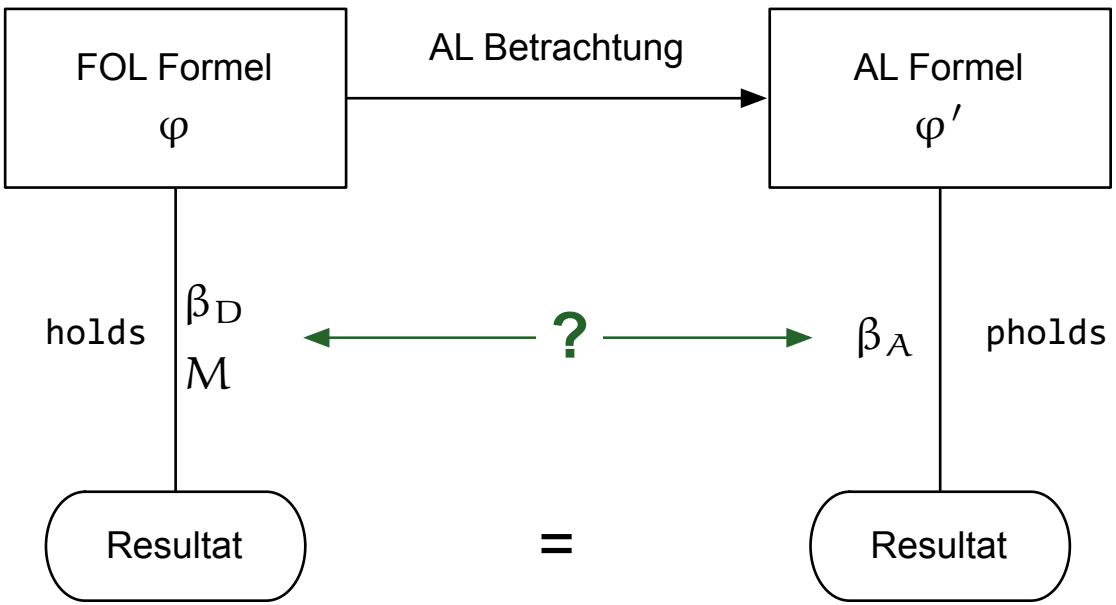
FOL Validität vs. AL Validität

Was wir zeigen wollen?

Eine quantorenfreie FOL Formel ist genau dann allgemeingültig, wenn die entsprechende aussagenlogische Betrachtung der Formel ebenfalls allgemeingültig ist.

Wie erreichen wir das?

Wir stellen eine Korrespondenz zwischen einer FOL Interpretation und Belegung und einer aussagenlogischen Belegung her.



Korrespondenz—1

$$\beta_D, M \to \beta_A$$

- β_D und M definieren bereits eine entsprechende aussagenlogische Belegung

$$\beta_A(R(t_1, \ldots, t_n)) = \mathtt{holds}(M, \beta_D, R(t_1, \ldots, t_n))$$

Theorem

Sei

- β_A *nach obigem Muster definiert*
- φ *eine quantorenfreie FOL Formel*

dann gilt für alle Interpretationen M *und alle Belegungen* β_D

$$\mathtt{pholds}(\beta_A, \varphi) = \mathtt{holds}(M, \beta_D, \varphi)$$

Theorem

Wenn die AL Betrachtung φ' *einer quantorenfreien FOL Formel* φ *eine aussagenlogische Tautologie ist, so ist* φ *gültig.*

Korrespondenz—2

$$\beta_A \to \beta_D, \mathcal{M}$$

- Aus einer AL Belegung β_A wollen wir eine Interpretation $\mathcal{M}$ und Belegung β_D erzeugen, so dass gilt

$$\texttt{holds}(\mathcal{M}, \beta_D, \varphi) = \texttt{pholds}(\beta_A, \varphi)$$

- Vorgehen etwas technischer, aber der wichtige Punkt:

Kanonische Interpretation

Wir können aus einer AL Belegung β_A eine Interpretation $\mathcal{M}_{\beta_A}$ (kanonische Interpretation) und eine Belegung β_D ableiten, so dass gilt:

$$\texttt{holds}(\mathcal{M}_{\beta_A}, \beta_D, \varphi) = \texttt{pholds}(\beta_A, \varphi)$$

Theorem

Eine quantorenfreie FOL Formel φ gültig, gdw. die AL Betrachtung φ' eine Tautologie ist.

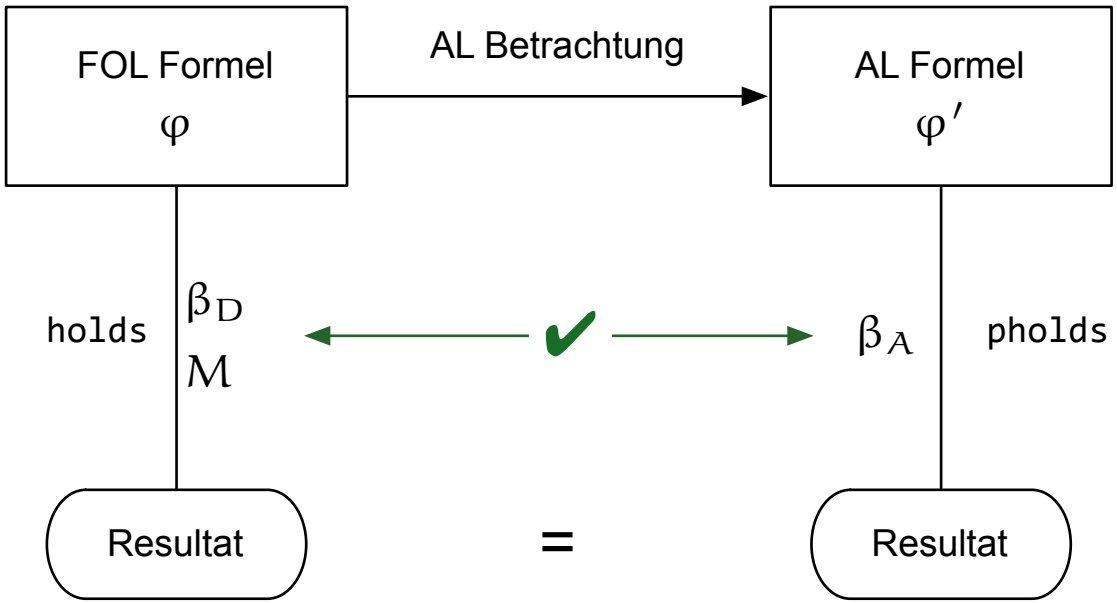
Was haben wir erreicht?

Was wir zeigen wollen?

Eine quantorenfreie FOL Formel ist genau dann allgemeingültig, wenn die entsprechende aussagenlogische Betrachtung der Formel ebenfalls allgemeingültig ist.

Wie erreichen wir das?

Wir stellen eine Korrespondenz zwischen einer FOL Interpretation und Belegung und einer aussagenlogischen Belegung her.



FOL Erfüllbarkeit vs. AL Erfüllbarkeit—1

- Die Korrespondenz zwischen FOL Validität und AL Validität ist interessant, aber:
  - Gilt nur für quantorenfreie Formeln
  - Skolemisierung (für quantorenfreie Formeln) ist nur erfüllbarkeits-erhaltend, nicht validitäts-erhaltend

i Variablenfreier Fall ist einfach!

Eine variablenfreie FOL Formel ist erfüllbar gdw. ihre AL Betrachtung erfüllbar ist.

Hinrichtung ist wieder einfach:

Theorem

Wenn eine quantorenfreie FOL Formel φ erfüllbar ist, so ist auch ihre AL Betrachtung φ' erfüllbar.

Wäre φ' nicht erfüllbar, so wäre $\neg\varphi'$ eine Tautologie. Damit wäre $\neg\varphi$ eine Tautologie und φ könnte nicht erfüllbar sein.

FOL Erfüllbarkeit vs. AL Erfüllbarkeit—2

Wir haben gezeigt: FOL Formel erfüllbar $\rightarrow$ AL Betrachtung erfüllbar

⚠ Umkehrrichtung ist nicht so einfach

Aussagenlogische Betrachtung von $P(x) \wedge \neg P(y)$ ist erfüllbar, nicht jedoch die original FOL Formel.

Wir gehen wieder den technischen Umweg über kanonische Interpretationen und daraus folgend kanonische Modelle.

Theorem

- φ *eine quantorenfreie FOL Formel*
- β_A *eine aussagenlogische Belegung von atomaren Formeln*
- $\mathcal{M}$ *eine kanonische Interpretation für* φ *mit* $P_{\mathcal{M}}(t_1, \ldots, t_n) = \beta_A(P(t_1, \ldots, t_n))$

dann gilt für eine beliebige Belegung β_D*:*

$$\texttt{holds}(\mathcal{M}, \beta_D, \varphi) = \texttt{pholds}(\beta_A, \texttt{subst}(\beta_D, \varphi))$$

Und damit: AL Betrachtung erfüllbar $\rightarrow$ FOL Formel erfüllbar

Herbrand-Universum

Idee!

Finde ein kanonisches Modell mit kleinst-möglichem Universum.

Definition (Herbrand-Universum)

Das Herbrand-Universum zu einer FOL Sprache ist die Menge aller variablenfreien Terme dieser Sprache.

- Wenn die Sprache keine Konstanten hat, wird eine Konstante c hinzugefügt.
- Herbrand-Universum (HU) für eine Formel φ: HU für die Sprache aus φ

Beispiel (Herbrand-Universum)

- Sprache: $\{P^{(1)}, R^{(3)}, a^{(0)}, f^{(1)}\}$
- HU$(P(x) \wedge R(x,y,z) \vee \neg P(y)) = \{c\}$
- HU$(P(a) \wedge R(x,y,z) \vee \neg P(f(y))) = \{a, f(a), f(f(a)), \ldots\}$

Herbrand-Interpretation

Definition (Herbrand-Interpretation)

Eine Herbrand-Interpretation ist eine kanonische Interpretation, deren Definitionsbereich das Herbrand-Universum ist.

D.h. eine Herbrand-Interpretation ordnet

- jedem Funktionssymbol f^n eine Funktion $f : HU^n \to HU$ zu

Beispiel (Herbrand-Interpretation)

- Sprache: $\{P^{(1)}, R^{(3)}, a^{(0)}, f^{(1)}\}$

$$\varphi = P(a) \wedge R(x, y, z) \vee \neg P(f(y))$$

- Herbrand-Universum

$$HU(\varphi) = \{a, f(a), f(f(a)), \ldots\}$$

- Herbrand-Interpretation
  - $a_M = a$
  - $f_M(t) = f(t)$

Herbrand-Modell und Grund-Instanzen

Definition (Herbrand-Modell)

Ein Modell einer Menge Γ von Formeln, das eine Herbrand-Interpretation ist, heißt auch Herbrand-Modell.

Definition (Grund-Instanz)

Eine variablenfreie Formel $\varphi\sigma$ heißt Grund-Instanz von φ, wenn σ in das Herbrand-Universum abbildet.

Beispiel (Grund-Instanzen)

- Sprache: $\{P^{(1)}, R^{(3)}, a^{(0)}, f^{(1)}\}$

$$\varphi = P(a) \wedge R(x, y, z) \vee \neg P(f(y))$$

Mögliche Grund-Instanzen

- $P(a) \wedge R(a, a, a) \vee \neg P(f(a))$
- $P(a) \wedge R(f(a), a, a) \vee \neg P(f(a))$
- $P(a) \wedge R(a, f(a), a) \vee \neg P(f(f(a)))$
- $\ldots$

Der Satz von Herbrand

Theorem (Erfüllbarkeit in Herbrand-Modellen)

Eine Herbrand-Interpretation $\mathfrak{H}$ erfüllt eine quantorenfreie Formel φ gdw. sie die Menge aller Grund-Instanzen von φ erfüllt.

Dieses Ergebnis gilt nicht nur für die Erfüllbarkeit in einem speziellen Herbrand-Modell, sondern ganz allgemein für die Erfüllbarkeit:

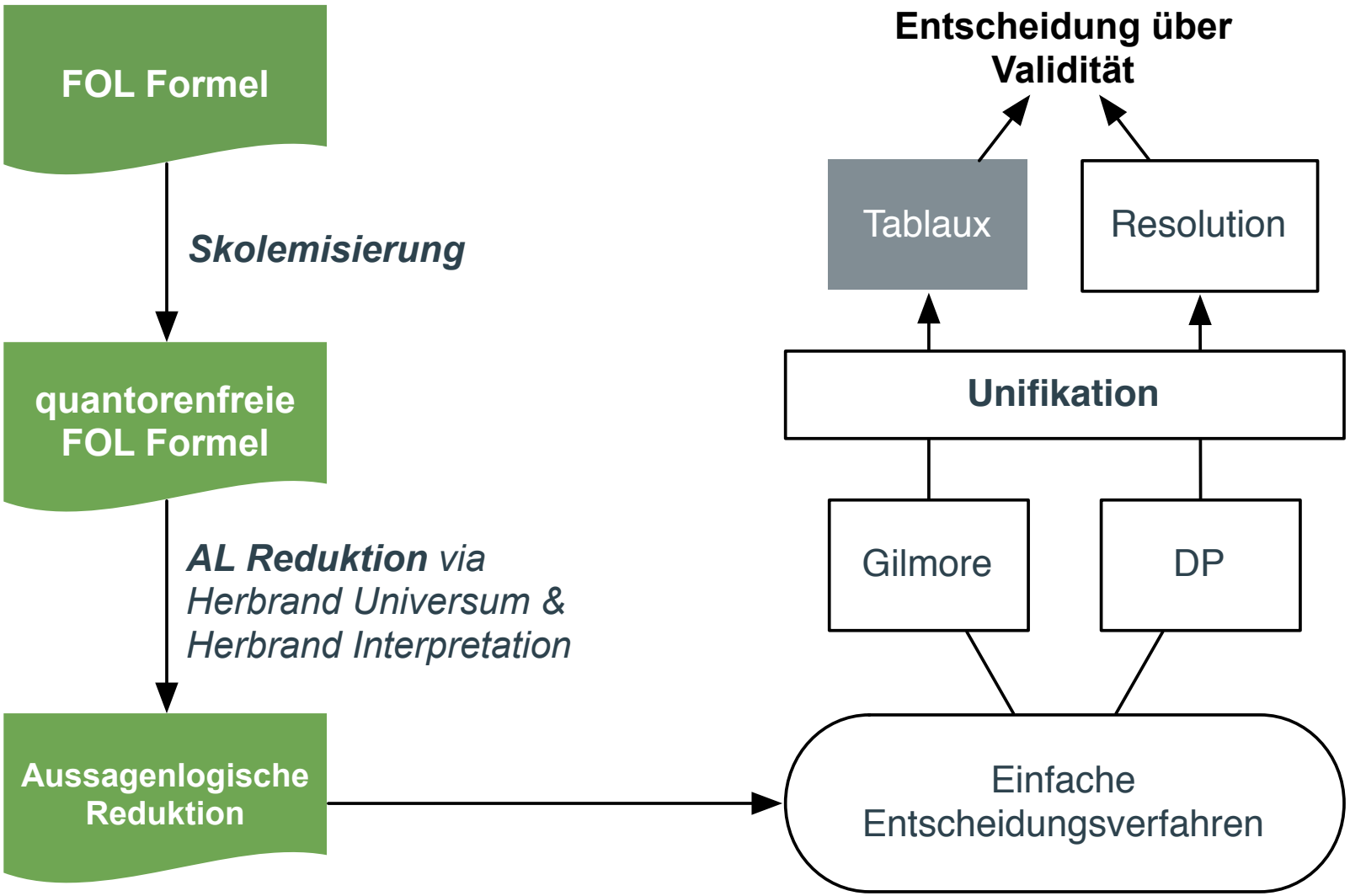
Theorem (Satz von Herbrand)

Eine quantorenfreie Formel φ ist erfüllbar in FOL, gdw. die Menge aller Grund-Instanzen aussagenlogisch erfüllbar ist.

Theorem

Eine quantorenfreie Formel hat ein Modell (d.h. ist erfüllbar) gdw. sie ein Herbrand-Modell besitzt.

Wo stehen wir?



Und wie implementiert man das?

Um die Erfüllbarkeit einer FOL Formel φ zu testen, können wir prinzipiell wie folgt vorgehen:

1. Bringe φ in Skolemform (und damit quantorenfrei bzw. rein universell quantifiziert)
2. Überprüfe, ob die Menge aller Grund-Instanzen aussagenlogisch erfüllbar ist

⚠ Problem!

Im Allgemeinen gibt es unendlich viele Grund-Instanzen!

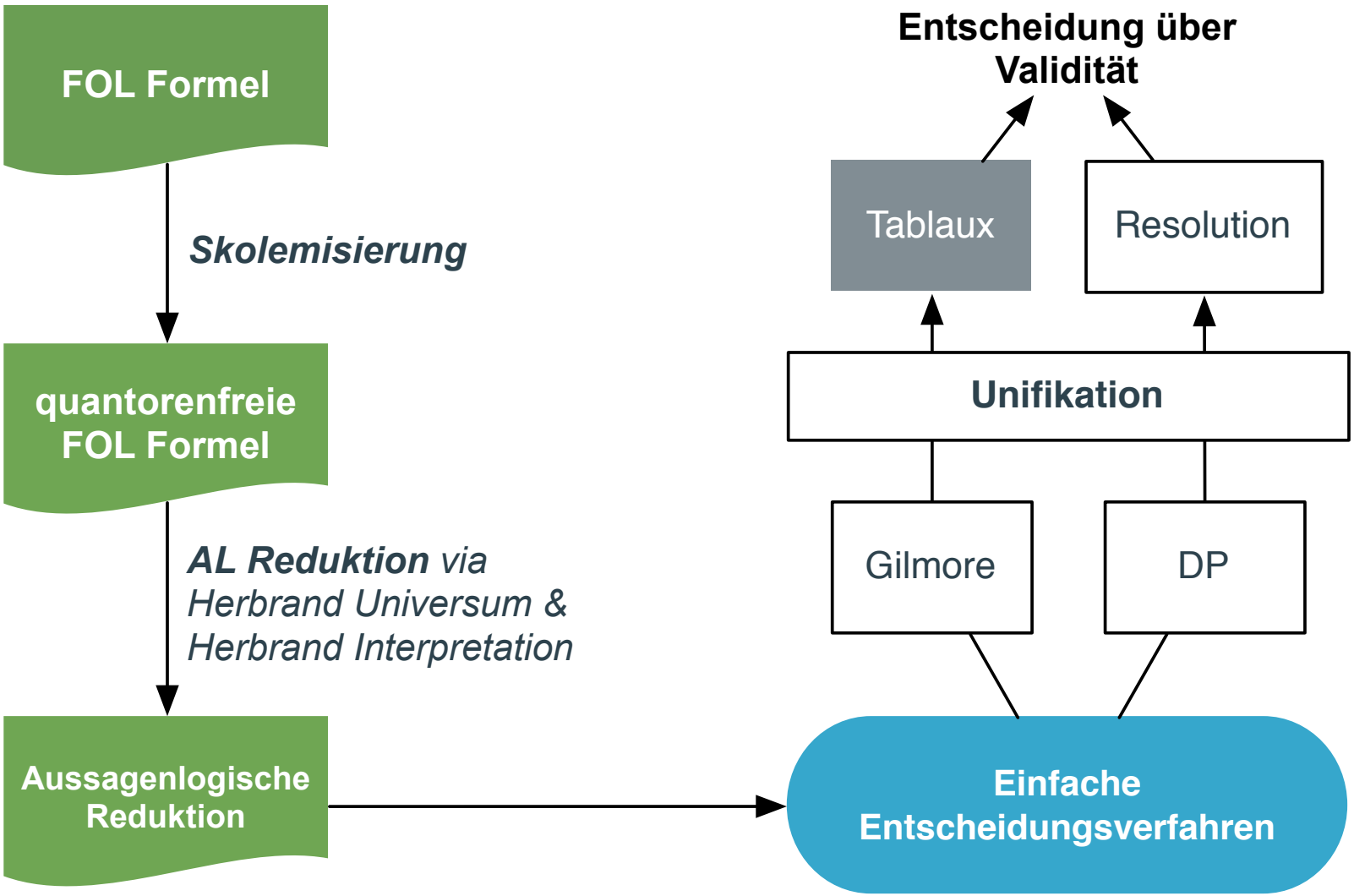
Idee!

Wende Kompaktheitssatz an: *Eine quantorenfreie FOL Formel ist erfüllbar, gdw. alle endlichen Mengen von Grund-Instanzen aussagenlogisch erfüllbar sind.*

Corollary (Unerfüllbarkeit einer FOL Formel)

Eine quantorenfreie FOL Formel φ ist unerfüllbar, gdw. eine beliebige endliche Menge von Grund-Instanzen aussagenlogisch unerfüllbar ist.

Wo stehen wir?



Ein erster Algorithmus von Gilmore (1960)

Grundidee!

1. Zähle immer größere Mengen von Grund-Instanzen auf
2. Teste die Konjunktion dieser Instanzen auf Unerfüllbarkeit

Verfahren von Gilmore (1960):

- In DNF konvertieren und komplementäre Literale suchen

Beispiel (Gilmores Verfahren)

- $\varphi = \exists x \forall y [P(x) \rightarrow P(y)]$
- Um Validität zu zeigen: $\varphi' = \neg \exists x \forall y [P(x) \rightarrow P(y)]$
- Skolemisierung: $\varphi'' = P(x) \wedge \neg P(sk_0(x))$
- 1. Grundinstanz: $i_1 = P(c) \wedge \neg P(sk_0(c))$ erfüllbar
- 2. Grundinstanz: $i_2 = P(sk_0(c)) \wedge \neg P(sk_0(sk_0(c)))$ erfüllbar, aber $i_1 \wedge i_2$ unerfüllbar

$\Rightarrow$ endliche Menge von Grundinstanzen ist unerfüllbar $\Rightarrow$ φ ist valide

Eine Verbesserung: Davis & Putnam (1960)

⚠ Problem bei Gilmore

- Man muss eine Konjuntion von Grund-Instanzen untersuchen
- DNF Konversion einer Konjunktion kann leicht explodieren

Bessere Idee!

Teste CNF auf Erfüllbarkeit, dann muss nur jede Grundinstanz selber in CNF konveritert werden, nicht jedoch das gesamte System.

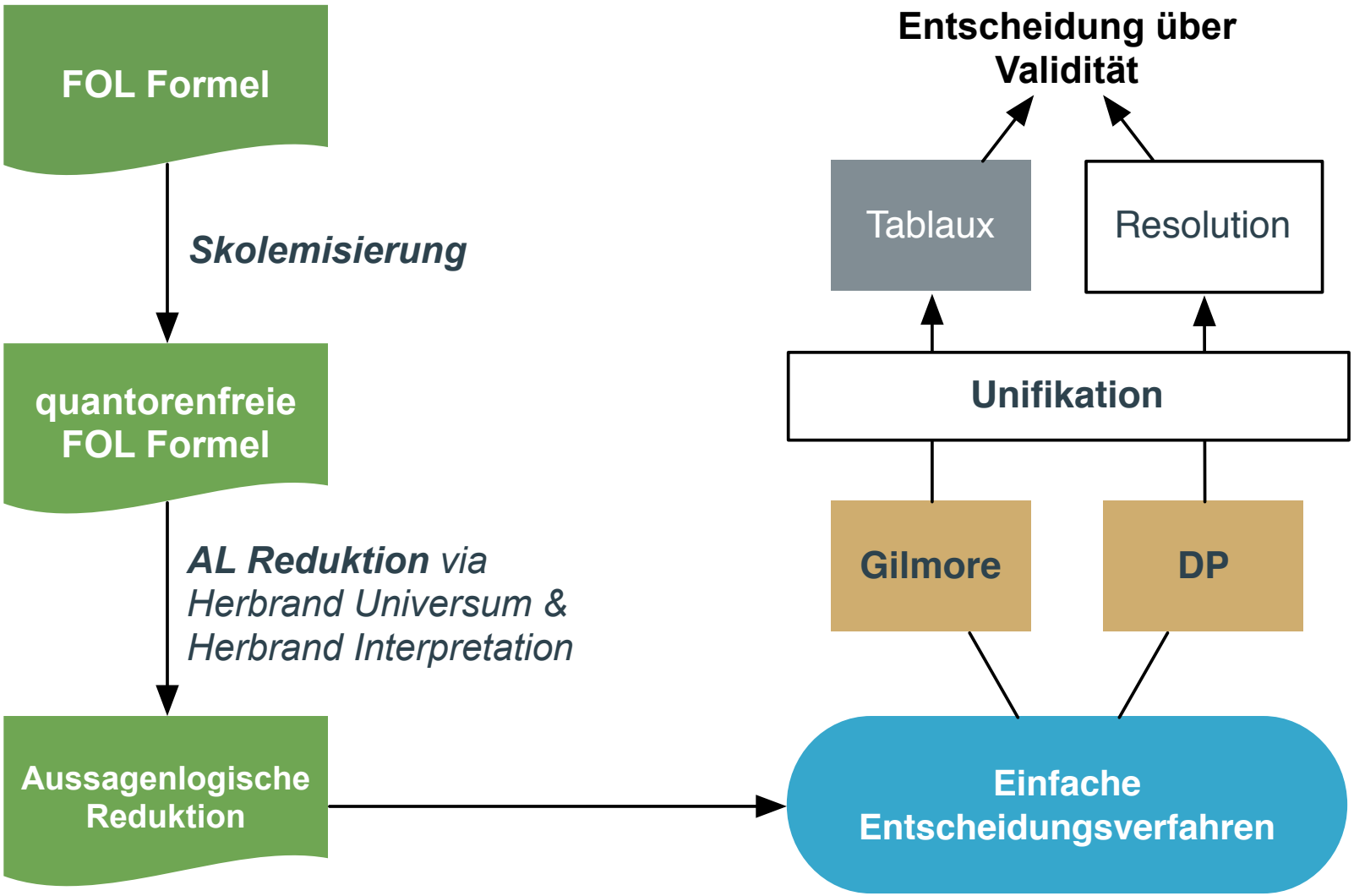
Dies war der ursprüngliche Zweck der Davis Putnam Prozedur:

1. Zähle immer größere Mengen an Grundinstanzen auf
2. Überprüfe mit dem DP(LL) Verfahren, ob die entstehende Konjunktion erfüllbar ist oder nicht

i Vergleich

DP erzielt in der Praxis meist sehr viel bessere Ergebnisse als das Verfahren von Gilmore.

Wo stehen wir?



Motivation für die Unifikation

- DP produziert immer noch sehr viele Grund-Instanzen
- Die meisten dieser Instanzen tragen nicht zur schlussendlichen Widerlegung der Aussage bei

$\Rightarrow$ Wähle Grundinstanzen klüger aus um möglichst schnell eine unerfüllbare Menge an Instanzen zu erzeugen

Idee!

Instanziiere Formeln intelligenter um AL Schlussfolgerungen zu treffen.

Beispiel (Instanziierung)

- Zwei uninstanziierte Klauseln: $P(x, f(y)) \vee Q(x, y)$ und $\neg P(g(u), v)$
- "Kluge" Instanziierung: $x \mapsto g(u)$ und $v \mapsto f(y)$
- Nach der Instanziierung: $P(g(u), f(y)) \vee Q(g(u), y)$ und $\neg P(g(u), f(y))$
- Mit AL Resolution kann nun eine neue Klausel $Q(g(u), y)$ geschlussfolgert werden

Unifikator

Definition (Unifikator)

Gegeben eine Menge von Paaren von Termen

$$S = \{(s_1, t_1), \ldots, (s_n, t_n)\}.$$

Ein Unifikator von S ist eine Substitutiton/Instanziierung σ, so dass

$$\texttt{tsubst}(\sigma, s_i) = \texttt{tsubst}(\sigma, t_i)$$

für alle $i = 1, \ldots, n$.

Spezialfall:

- Nur ein Termpaar in $S = \{(s, t)\}$: Unifikator von s und t

Beispiel (Unifikator)

- $S = \{(x, g(u)), (f(y), v)\}$
- Unifikator $\sigma = \{x \mapsto g(u), v \mapsto f(y)\}$

Analogie: Lineares Gleichungssystem

Unifikation ist vergleichbar mit Lösen eines linearen Gleichungssystems

	Unifikation	**LGS**
Lösbar	$\{(x, g(u)), (f(y), v)\}$	$2x + y = 3$ $x - y = 6$
Unlösbar	$\{g(x), f(y)\}$	$x = 7$ $x = -3$
Occurence	$\{x, f(x)\}$	$x = x + 1$
Zirkularität	$\{(x, f(y)), (y, g(x))\}$	$x = y + 1$ $y = x + 2$

Unifikationsproblem lösbar $\Rightarrow$ immer unendlich viele Lösungen.

Beispiel (Verschiedene Unifikatoren)

$S = \{(x, g(u)), (f(y), v)\}$

- $\sigma_0 = \{x \mapsto g(u), v \mapsto f(y)\}$
- $\sigma_1 = \{x \mapsto g(f(g(y))), u \mapsto f(g(y)), v \mapsto f(y)\}$
- ...

Most General Unifier (MGU)

Definition (allgemein)

Eine Instanziierung σ ist allgemeiner als eine andere τ, $\sigma \leqslant \tau$, wenn es eine weitere Instanziierung δ gibt mit

$$\texttt{tsubst}(\tau, t) = \texttt{tsubst}(\delta, \texttt{tsubst}(\sigma, t))$$

Beispiel (Unifikatoren)

$S = \{(x, g(u)), (f(y), v)\}$

- $\sigma_0 = \{x \mapsto g(u), v \mapsto f(y)\}$
- $\sigma_1 = \{x \mapsto g(f(g(y))), u \mapsto f(g(y)), v \mapsto f(y)\}$

$\sigma_0 \leqslant \sigma_1$ mit $\delta = \{u \mapsto f(g(y))\}$

Definition (Allgemeinster Unifikator)

σ ist ein Allgemeinster Unifikator (Most General Unifier, MGU) von S, wenn

1. σ ein Unifikator von S ist und
2. für alle anderen Unifikatoren τ von S gilt, dass $\sigma \leqslant \tau$.

Der Algorithmus—Anforderungen

Eingabe:

- Menge an Termpaaren: $S = \{(s_1, t_1), \ldots, (s_n, t_n)\}$
- Γ: Abbildung von Variablen zu Termen

Vorgehen:

- Versuche, die Termpaare in S zu unifizieren und
- Speichere die entstehenden Abbildungen von Variablen zu Termen in Γ

⚠ Achtung!

Γ ist noch nicht der finale Unifikator, da hier noch Substitutionsketten enthalten sein können, z.B. $x \mapsto y$ und $y \mapsto z$. Ein Unifikator ist jedoch eine simultane Substitution.

- Wir müssen sicherstellen, dass Γ keine Zyklen enthält.

Beispiel (Zyklus)

$\Gamma = \{x \mapsto f(y, z), y \mapsto w, w \mapsto x\}$

Der Algorithmus

Algorithmus: $\texttt{unify}(\Gamma, S)$

Eingabe: Abbildung von Variablen zu Termen Γ, Menge von Termpaaren, die unifiziert werden sollen $S = \{(s_1, t_1), \ldots, (s_n, t_n)\}$
Ausgabe: Abbildung von Variablen zu Termen, die zu einem Unifkator von S angewandt werden können, oder $\bot$, falls Unifkation nicht möglich

---

$\texttt{unify}(\Gamma, S) = S$ *match*

$\emptyset \rightsquigarrow \Gamma$

| $\{(s,t)\} \cup S' \rightsquigarrow$ mit $s = t \rightsquigarrow \texttt{unify}(\Gamma, S')$

| $\{(f(s_1, \ldots, s_n), g(t_1, \ldots, t_m))\} \cup S' \rightsquigarrow$ *if* $f = g \wedge n = m$ *then* $\texttt{unify}(\Gamma, \{(s_1, t_1), \ldots, (s_n, t_m)\} \cup S')$ *else* $\bot$

| $\{(x,t)\} \cup S'$ mit $x \in \mathcal{V}_T \rightsquigarrow$ *if* $x \mapsto s \in \Gamma$ *then* $\texttt{unify}(\Gamma, \{(s,t)\} \cup S')$ *else* $\texttt{unify}(\Gamma \cup \{x \mapsto t\}, S')$ *(mit Zyklus Check)*

| $\{(t,x)\} \cup S'$ mit $x \in \mathcal{V}_T \rightsquigarrow \texttt{unify}(\Gamma, \{(x,t)\} \cup S')$

Der Algorithmus—Beispiel

Beispiel (unify)

$$S = \{[f(x, h(y)), f(g(a), z)], [z, h(y)], [w, z]\}$$

$$\begin{aligned}
\texttt{unify}(\emptyset, S) &= \texttt{unify}(\emptyset, \{[x, g(a)], [h(y), z], [z, h(y)], [w, z]\}) \\
&= \texttt{unify}(\{x \mapsto g(a)\}, \{[h(y), z], [z, h(y)], [w, z]\}) \\
&= \texttt{unify}(\{x \mapsto g(a)\}, \{[z, h(y)], [z, h(y)], [w, z]\}) \\
&= \texttt{unify}(\{x \mapsto g(a), z \mapsto h(y)\}, \{[z, h(y)], [w, z]\}) \\
&= \texttt{unify}(\{x \mapsto g(a), z \mapsto h(y)\}, \{[h(y), h(y)], [w, z]\}) \\
&= \texttt{unify}(\{x \mapsto g(a), z \mapsto h(y)\}, \{[w, z]\}) \\
&= \texttt{unify}(\{x \mapsto g(a), z \mapsto h(y), w \mapsto z\}, \{\}) \\
&= \{x \mapsto g(a), z \mapsto h(y), w \mapsto z\}
\end{aligned}$$

Achtung!

Funktion `unify` zeigt an, dass S unifizierbar ist, liefert aber noch keinen korrekten Unifikator: $\{[f(g(a)), h(y), f(g(a)), h(y)], [h(y), h(y)], [z, h(y)]\}$

Der Algorithmus—MGU

MGU aus Γ

Vorgehen:

- Wende Substitutionen in Γ bis zum Fixpunkt an

Beispiel (MGU aus Γ)

$$S = \{[f(x, h(y)), f(g(a)), z], [z, h(y)], [w, z]\}$$

$$\texttt{unify}(\emptyset, S) = \{x \mapsto g(a), z \mapsto h(y), w \mapsto z\}$$

MGU aus Γ:

$$\sigma = \{x \mapsto g(a), z \mapsto h(y), w \mapsto h(y)\}$$

$$S\sigma = \{[f(g(a)), h(y), f(g(a)), h(y)], [h(y), h(y)], [h(y), h(y)]\}$$

- Es kann aber klug sein, den MGU nicht direkt zu speichern, sondern nur Γ

Komplexität der Unifikation

MGU Berechnung im worst-case exponentiell in Zeit und Platz.

Beispiel (MGU im worst-case)

$t = f(x_1, x_2, \ldots, x_n)$ und $s = f(g(x_0, x_0), g(x_1, x_1), \ldots, g(x_{n-1}, x_{n-1}))$

- Beide Terme hängen linear von n ab
- $|O(t)| = n + 1$
- $|O(s)| = 3n + 1$
- Damit haben wir insgesamt $4n + 2$ Symbole

Der resultierende MGU σ hat die Gestalt:

$$\begin{array}{ccccc} x_1 & & x_2 & & \\ \downarrow & & \downarrow & & \\ g(x_0, x_0) & & g(g(x_0, x_0), g(x_0, x_0)) & & \ldots \end{array}$$

- D.h. für x_i wird Term der Größe $2^{i+1} - 1$ substituiert

$\Rightarrow |O(t\sigma)| = 1 + \sum_{i=1}^{n} \left(2^{i+1} - 1\right) = 2^{n+2} - n$ *(und damit exponentiell)*

Ansätze für gute Unifikationsalgorithmen

1. Speichern von Γ anstelle von MGU σ
2. Eindeutige Darstellung von Variablen, die an mehreren Positionen auftreten
3. Geschickte Reihenfolge für die Auswahl des nächsten Term-Paares

Unifikation nach Martelli und Montanari (1982)

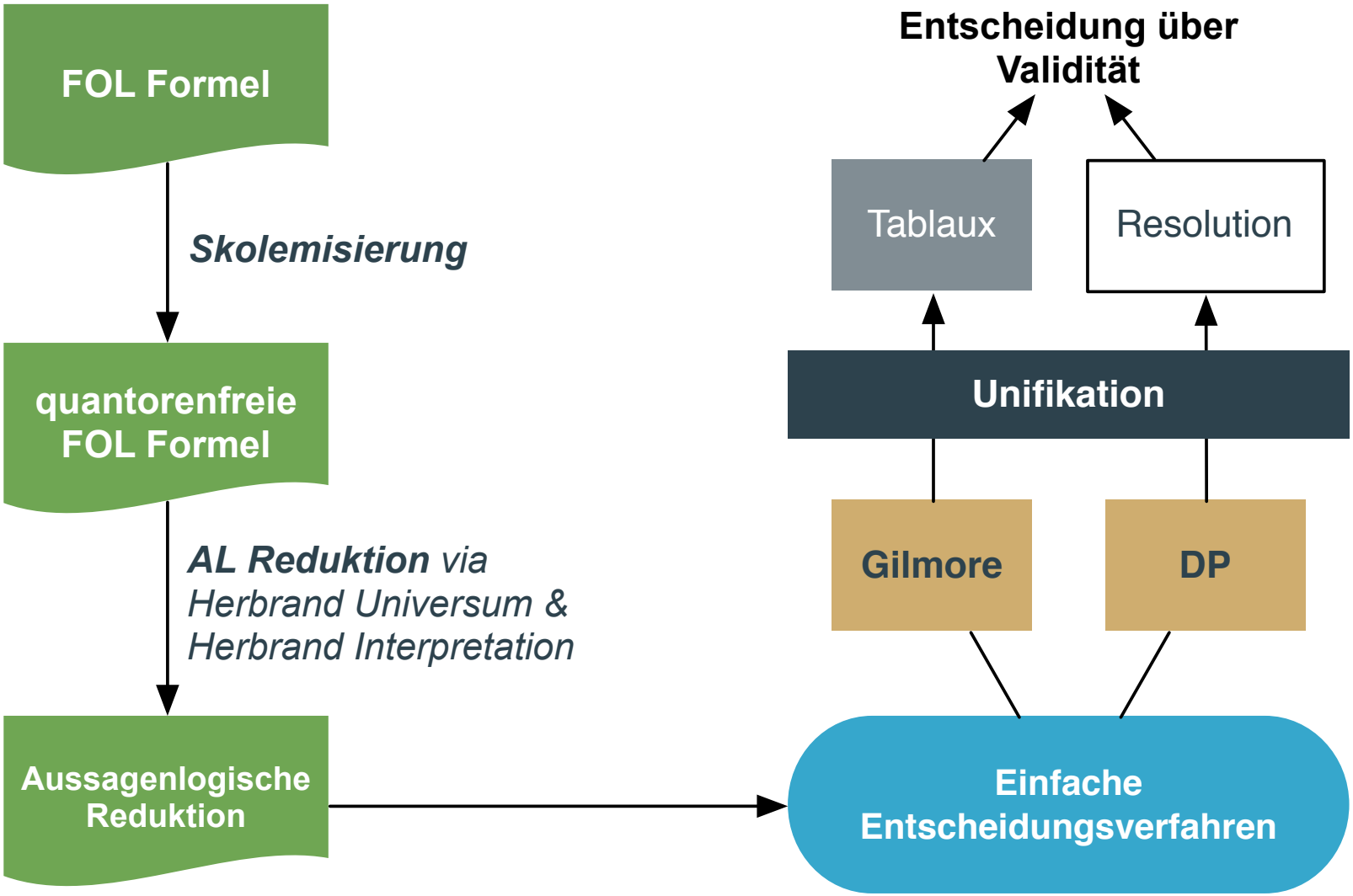
Gleichungsbasierter Unifikationsalgorithmus mit Komplexität $O(n \log n)$

Literaturhinweis

Ein guter Überblick über verschiedene Unifikationsalgorithmen findet sich in:

Franz Baader, Tobias Nipkow. **Term Rewriting and All That.** Cambridge University Press, 1999.

Wo stehen wir?



Wiederholung: Resolution

Wir können eine aussagenlogische Variable x via Resolution eliminieren:

1. Berechne alle möglichen Resolventen von Klauseln, die x oder $\neg x$ enthalten und füge sie hinzu
2. Lösche alle Klauseln, die x oder $\neg x$ enthalten

Entstehende Formel bleibt erfüllbarkeitsäquivalent

Beispiel (Elimination von x und y)

$\varphi = (x \vee \neg w) \wedge (\neg x \vee z) \wedge (\neg z \vee y) \wedge (w \vee z)$

1. y kommt nur in einer Phase vor $\Rightarrow$ Keine Resolventen möglich, kann direkt gelöscht werden

$$(x \vee \neg w) \wedge (\neg x \vee z) \wedge (w \vee z)$$

2. Alle Resolventen mit x berechnen

$$\texttt{resolvent}((x \vee \neg w), (\neg x \vee z)) = (\neg w \vee z)$$

3. Resolvente hinzufügen und alle Originalklauseln löschen

$$(w \vee z) \wedge (\neg w \vee z) \equiv z$$

Motivation für First-Order Resolution

- Satz von Herbrand: CNF einer FOL Formel unerfüllbar, gdw. eine endliche Teilmenge der aussagenlogischen Instanziierung unerfüllbar ist
- Aussagenlogische Unerfüllbarkeit kann mit Resolution gezeigt werden
- J. A. Robinson, 1965: Wende Unifikation an, so dass die "allgemeinsten" Formen der Klauseln direkt resolviert werden

Resolutionsregel für die FOL

Resolution

$$\frac{(\varphi \vee P) \wedge (\psi \vee \neg P)}{(\varphi \vee \psi\mu)\sigma}$$

- μ: Umbenennung, die die Variablen von φ und ψ trennt
- σ : MGU von P und $\neg P\mu$

Wir haben ein Problem

⚠ Problem!

Mit der Resolutionsregel alleine ist der Kalkül nicht widerlegungsvollständig:

$$\varphi = (P(x) \vee P(y)) \wedge (\neg P(u) \vee \neg P(v))$$

- Alle entstehenden Resolventen haben wieder 2 Literale
- Leere Klausel wird nie erreicht
- φ ist aber unerfüllbar (Betrachte Fall $x = y = u = v$)

Idee

- Das Problem kommt daher, dass in einer Klausel das selbe Prädikat mehr als einmal vorkommt

$\Rightarrow$ Ersetze alle Vorkommen des Prädikats durch die allgemeinste Form des Prädikats

Die Faktorisierungsregel

$$\frac{\varphi \vee P(t_{11}, \ldots, t_{1n}) \vee \cdots \vee P(t_{m1}, \ldots, t_{mn})}{(\varphi \vee P(t_{11}, \ldots, t_{1n}))\sigma}$$

mit σ MGU von $P(t_{11}, \ldots, t_{1n}), \ldots, P(t_{m1}, \ldots, t_{mn})$

Beispiel (Resolution mit Faktorisierung)

$$\varphi = (P(x) \vee P(y)) \wedge (\neg P(u) \vee \neg P(v))$$

Nach Faktorisierung:

$$\varphi' = \cdots \wedge P(x) \wedge \neg P(u)$$

Nach einem Resolutionsschritt: leere Klausel.

Definition (Volle Resolution nach J. A. Robinson, 1965)

$$\frac{(\varphi \vee P(t_{11}, \ldots) \vee P(t_{m1}, \ldots)) \wedge (\psi \vee \neg P(s_{11}, \ldots) \vee \neg P(s_{k1}, \ldots))}{(\varphi \vee \psi\mu)\sigma}$$

- μ: Umbenennung, die die Variablen von Klausel 1 und 2 trennt
- σ: MGU von $P(t_{11}, \ldots), \ldots P(t_{m1}, \ldots), P(s_{11}, \ldots)\mu, \ldots P(s_{k1}, \ldots)\mu$

Ein komplettes Beispiel—1

Beispiel (Resolution)

$$\forall y \forall z\ ((P(y, f(z)) \vee P(f(z), y)) \wedge \neg P(f(x), z))$$

Skolemisierung & Klauselform:

$$\{\{P(y, f(z)), P(f(z), y)\}, \{\neg P(f(x), z)\}\}$$

Anwendung der Faktorisierungsregel auf Klausel 1:

$$\{\{P(f(z), f(z))\}, \{\neg P(f(x), z)\}\}$$

Trennen der Variablen

$$\{\{P(f(y), f(y))\}, \{\neg P(f(x), z)\}\}$$

Berechnen des MGU:

$$\sigma = \{x \mapsto y, z \mapsto f(y)\}$$

Resolvente:

$$\{\}$$

Ein komplettes Beispiel—2

Beispiel (Resolution)

Sprache:

$$(b^{(0)}, c^{(0)}, g^{(1)}, f^{(1)}, h^{(1)}, Q^{(1)}, P^{(2)})$$

Klauselmenge $\{c_1, c_2, c_3, c_4\}$ mit

$$c_1 = \{\neg Q(z)\},$$

$$c_2 = \{\neg P(x, g(b))\},$$

$$c_3 = \{Q(c), P(c, g(z)), Q(z)\},$$

$$c_4 = \{Q(c), P(c, g(z)), P(f(h(z)), y)\}$$

Resolvente von Klausel c_1 und c_3:

$$\frac{\neg Q(z) \wedge (Q(c) \vee P(c, g(z)) \vee Q(z))}{(P(c, g(z))\mu)\sigma}$$

- $\mu = \{z \mapsto w\}$
- $\sigma = \{z \mapsto c, w \mapsto c\}$: MGU von $Q(z), Q(c), Q(w)$

$c_5 = P(c, g(c))$ . . . Tafel!

Das Lifting Lemma

- Literalmenge C: $C^{-} = \{\neg p \mid p \in C\}$

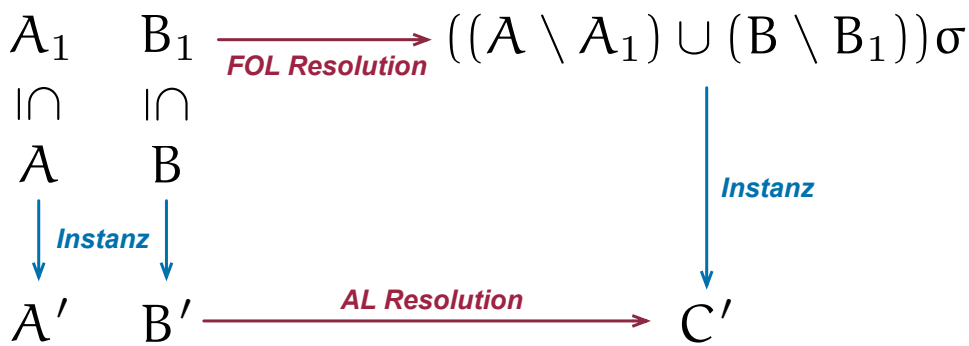
Lemma (Lifting Lemma)

Seien

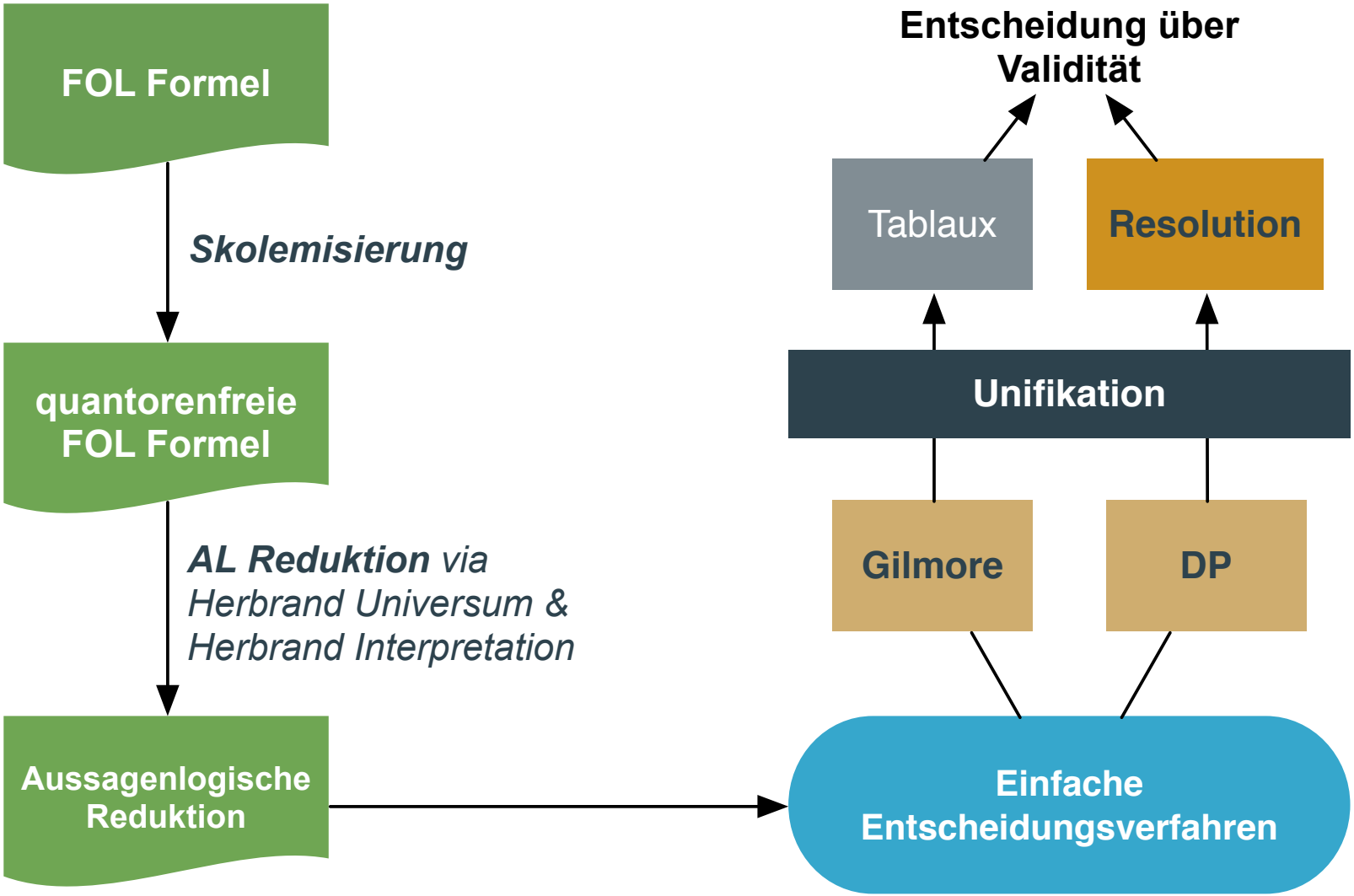
- A *und* B *FOL-Klauseln mit* $\text{vars}(A) \cup \text{vars}(B) = \emptyset$
- A' *und* B' *Instanzen von* A *und* B*, so dass* A' *und* B' *die aussagenlogische Resolvente* C' *haben*

Dann gibt es nichtleere Teilmengen $A_1 \subseteq A$ *und* $B_1 \subseteq B$*, so dass*

- $S = A_1 \cup B_1^{-}$ *unifizierbar und*
- *für einen beliebigen MGU* σ *von* S *gilt:* C' *ist eine Instanz von* $((A \setminus A_1) \cup (B \setminus B_1))\sigma$



Endlich...


FOL Formel
Skolemisierung
quantorenfreie FOL Formel
AL Reduktion via Herbrand Universum & Herbrand Interpretation
Aussagenlogische Reduktion
Entscheidung über Validität
Tablaux
Resolution
Unifikation
Gilmore
DP
Einfache Entscheidungsverfahren

Literaturtipps

Literaturhinweis

- *John Harrison.* **Chapter 3—First Order Logic in Handbook of Practical Logic and Automated Reasoning.** Cambridge University Press, 2009.
- *Leo Bachmair & Harald Ganzinger.* **Chapter 1—Resolution Theorem Proving in Handbook of Automated Reasoning.** Elsevier, 2001.
- *Dieter Hofbauer & Ralf-Detlef Kutsche.* **Kapitel 1—Grundlagen der Prädikatenlogik & 2—Resolution in Grundlagen des maschinellen Beweisens.** Vieweg, 1991.

Web Links

- `http://www.cs.miami.edu/~tptp/` — Sammlung von First-Order und Higher-Order Problem Instanzen
- `http://www.vprover.org/` — Sehr erfolgreicher FOL Theorem Prover von Andrei Voronkov
- `http://www.spass-prover.org/` — FOL Theorem Prover vom Max Planck Institut Saarbrücken