Fachhochschule Bielefeld
Campus Minden

Informatik

Maschinelle Überprüfung der Beweisführungen von logischen Ausdrücken

Forschungsprojekt

Wintersemester: 2020/2021

vorgelegt von: David Nickel
Matrikelnummer: 1111275

Abgabe am: 15.04.2021

Prüfer: Dipl.-Inform B. C. George

1 Inhalt

2 Einleitung

Es ist eine mühsame und langwierige Angelegenheit logische Beweise auf Korrektheit zu überprüfen. In dem Modul „Formal Models in Computer Science“ in dem Masterstudiengang Informatik an der FH Bielefeld am Campus Minden, werden diese logischen Beweise betrachtet. In der Modulprüfung müssen die Studenten Beweise ausarbeiten, sodass der Dozent in die Verlegenheit kommt, diese überprüfen zu müssen. Wenn es möglich ist, die Beweise maschinell zu überprüfen, dann wird diese Verlegenheit eine gute Gelegenheit die Prüfungen zu automatisieren.

Ziel dieser Arbeit ist in erster Linie, die Machbarkeit einer maschinellen Überprüfung zu ermitteln. Dabei sollen auch eine Datenstruktur, eine Projektstruktur und wichtige Methoden entwickelt werden, um einen guten Lösungsansatz für die vollständige Implementierung zu bieten.

2.1 Vorlagen

In den nicht veröffentlichen Arbeiten „Ein sprachtheoretischer Ansatz zur automatischen Überprüfung von Berechnungen mithilfe eines Termersetzungssystems“ von Nils Dralle und „Entwicklung einer Frontendanwendung für ein Termersetzungssystem unter besonderer Berücksichtigung von Usabilty und Sicherheitsaspekten“ von Michael Nickel wurde eine Anwendung entwickelt, die dazu dient, mathematische Ausdrücke auf Gleichheit zu überprüfen. Dieses System dient als Vorlage.

2.2 Vorgehen

Zuerst soll der Eingabestring eingelesen werden.

Dann ausgewertet.

Dann das Ergebnis ausgegeben.

Für die logischen Ausdrücke und die Beweise werden alle Vorgaben aus dem Buch „Logic in Computer Science“[1] von Michael Huth und Mark Ryan übernommen.

Um die Beweise maschinell einlesen zu können, muss zuerst eine Grammatik entwickelt werden, die mächtig genug ist, diese abzubilden. Die Grammatik wird mit Antlr4 erstellt, ein entsprechender Parser, der aus der Eingabe einen Syntaxbaum erstellt, wird von Antlr generiert

Im nächsten Schritt muss die Information aus dem Syntaxbaum in eine geeignete Datenstruktur gespeichert werden. Dazu bietet es sich an, Visitoren zu implementieren, die den Syntaxbaum traversieren. Die Datenstruktur muss die Überprüfung der Beweise möglichst gut ermöglichen.

Wenn die Datenstruktur erstellt wurde, ist alles vorbereitet, um die tatsächliche Überprüfung der Beweise anzugehen.

Die letzten zwei Schritte sind die Schaffung einer Schnittstelle zu einem Frontend, damit das System getestet werden kann und das Testen mit möglichst vielen Beispielen.

[1] (Huth & Ryan, 2004)

3 Hauptteil

3.1 Interpretation der Eingabe

Um den Beweis maschinell einlesen zu können, muss eine Grammatik erarbeitet werden. Dazu wird Antlr4 verwendet. Es werden die Latex-Symbole verwendet.

Es wurde eine Grammatik entwickelt, die die logischen Beweise eindeutig darstellen kann. ein logischer Beweis ist in Blöcke gegliedert, welche wiederum in Zeilen unterteilt sind. Jede Zeile besteht aus der Zeilennummer, dem logischen Ausdruck und der Beweisregel, die angewendet wurde.

3.2 Entwicklung der Datenstruktur

Der Syntaxbaum muss in einen AST umgeformt werden, sodass die Informationen zur Überprüfung der Beweisregeln verwendbar sind.

Dazu werden verschiedene Knoten erstellt, die unterschiedliche Eigenschaften haben können. Die Datenstruktur ist im Anhang als Klassendiagramm dargestellt.

3.2.1 Block-Node

Jeder Beweis besteht aus einem oder mehreren Blöcken. Der gesamte Beweis ist in dem obersten Block. Dieser kann weitere Blöcke beinhalten und so weiter. Ein Block kann außerdem noch mehrere Zeilen beinhalten. Sowohl die Zeilen als auch die Blöcke werden als Kinder des oberen Blocks gespeichert. Jeder Block speichert zudem seine erste Zeilennummer und seine letzte. Bei der Prädikatenlogik können Variablen deklariert werden. Diese ebenfalls im Block-Node gespeichert.

3.2.2 Line-Node

Jede Zeile besteht aus einer Zeilennummer, aus einem logischen Ausdruck und aus einer Beweisregel. Am Anfang eines Blocks kann auch eine Variable deklariert werden. Alle diese Informationen sollen in dem Knoten gespeichert werden. Der logische Ausdruck und die Beweisregeln werden als Nodes angefügt.

3.2.3 Formula-Node

Ein Formula-Node besteht aus einem Operator und zwei Ausdrücken. Die Ausdrücke sind wiederum Nodes.

3.2.4 UnaryOperator-Node

Es gibt drei unäre Operatoren, die für die Beweise verwendet werden: Allquantor, Existquantor und Negation. Die Quantoren haben als zusätzliche Information eine Variable, die sie binden. In dem Node wird der Operator, die Variable (falls vorhanden) und der Ausdruck, auf den sich der Operator bezieht.

3.2.5 Predicate-Node

Prädikate werden im Predicate-Node gespeichert. Ein Prädikat kann Variablen als Parameter beinhalten.

3.2.6 Variable-Node

Variable-Node beschreibt die Variablen in den Ausdrücken

3.2.7 Bottom-Node

Bottom-Node beschreibt das Bottom-Symbol in den Ausdrücken

3.2.8 Rule-Node

Für jede Beweisregel gibt es eine Node, die die abstrakte Klasse RuleNode beerbt. Die Nodes für die Regeln unterscheiden sich in den Parametern, entsprechend den Zeilen oder Blöcken, die bei Verwendung der Regel angegeben werden müssen.

Der AST wird mit Visitoren aus dem Syntaxbaum erstellt, die predicate_proof_grammarBaseVisitor implementieren. predicate_proof_grammarBaseVisitor wird von Antlr aus der Grammatik generiert.

Für das Beispiel in Abbildung 1 wird die Datenstruktur in Abbildung 2 erstellt.

1	$p \rightarrow q$	premise
2	$\neg q$	assumption
3	$\neg p$	MT $1, 2$
4	$\neg q \rightarrow \neg p$	$\rightarrow$i 2–3

Abbildung 1: Beispiel (Huth & Ryan, 2004)

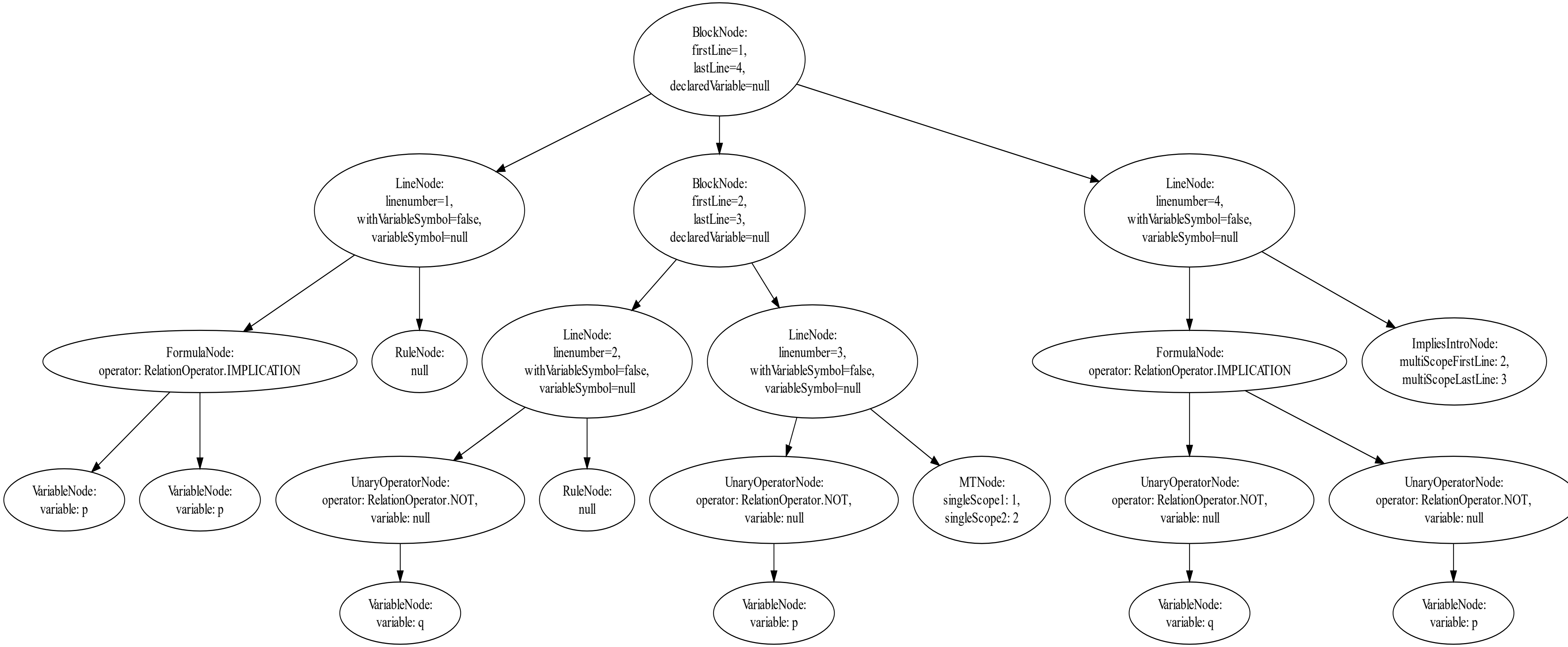


Abbildung 2: Beispieldatenstruktur

3.3 Überprüfung der Beweise

Der logische Beweis ist dann korrekt, wenn die Prämissen den Vorgaben entsprechen, das Ergebnis, der Erwartung entspricht und jeder logische Ausdruck auf eine korrekt angewendete Beweisregel zurückzuführen ist. Die Überprüfung der Prämissen und des Ergebnisses sind nicht Teil dieses Projekts. Ziel dieses Projekts ist es, die Überprüfung der korrekten Anwendung der Beweisregeln zu erforschen und zu implementieren. Dazu wurde für jeden Beweis eine Klasse in package „rule“ erstellt, die eine public-Methode „check“ enthält, welche einen boolean-Wert zurückliefert, ob die Regel korrekt angewandt wurde.

Der gesamte Beweis ist in einem äußeren Block gespeichert. In diesem Block können weitere Blöcke und Zeilen sein, die als children gespeichert werden. Jeder innere Block kann weitere Blöcke und Zeilen enthalten. In dieser Struktur wird rekursiv jeder Block und jede Zeile überprüft, angefangen bei dem obersten Block.

3.3.1 Regelbeschreibungen

Eine Beweisregel (Abbildung 3) besteht aus der Kennung (rote Markierung), welche Regel angewandt wurde und aus Verweisen auf vorherige Zeilen (grüne Markierung) oder Blöcke (blaue Markierungen). Entsprechend der Kennung werden die nodes erstellt und in dem LineNode gespeichert. Dies geschieht bereits im Visitor.

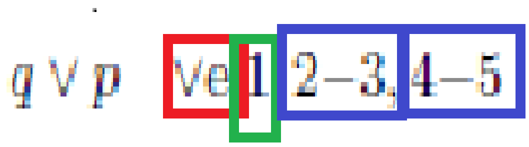


Abbildung 3: Reglebeschreibung (Huth & Ryan, 2004)

3.3.1.1 premise

Mit premise werden die Prämissen bezeichnet.

3.3.1.2 assumption

Mit assumption werden Annahmen bezeichnet. Annahmen werden zumeist am Beginn eines Blocks gemacht.

3.3.1.3 alredy proofed

Für die Beweise können auch bereits bewiesene Umformungen genutzt werden. Um das zu gewährleisten wird als Regel „already proofed“ eingeführt. Die Überprüfung wird in der Klasse AlreadyProofedRule durchgeführt. Bis jetzt ist nur „p → q ⊣⊢ ¬p ∨ q“ als already proofed programmiert.

3.3.1.4 copy

Wenn ein logischer Ausdruck bereits verwendet wurde in einem Beweise oder er eine Prämisse ist, kann er mit dem Schlüsselwort „copy“ wiederholt werden.

3.3.2 Algorithmen zur Überprüfung

Eine Schwierigkeit besteht darin, dass sich die Beweisregeln immer auf vorherige Zeilen oder Blöcke beziehen. Im AST sind die Zeilennummern gespeichert. Damit aber nicht jedesmal der ganze Baum durchsucht werden muss, werden alle bereits überprüften Zeilen oder Blöcke in einer Liste abgelegt. Die Funktionen searchLineNode und searchBlockNode aus CheckProof dursuchen anhand der übergebenen Zeilennummern die Listen und geben den entsprechenden Knoten zurück.

Im package rule ist für jede Beweisregel eine Klasse erstellt mit der Methode check.

Das Vorgehen ist immer ähnlich. Beispielhaft soll das Vorgehen bei der And-Introduction (Abbildung 4) erläutert werden.

$$\frac{\phi \quad \psi}{\phi \wedge \psi} \wedge \mathrm{i}$$

Abbildung 4: And-Introduction (Huth & Ryan, 2004)

Bei der Beweisregel für die And-Introduction muss der linke Operand des Ergebnisses gleich dem Ausdruck einer der angegebenen Zeilen sein und der rechte Operand gleich dem Ausdruck der anderen angegebenen Zeile.

In Rule wurden Methoden implementiert, die Knoten auf Gleichheit überprüfen. Dazu wird rekursiv vom obersten Knoten abwärts verglichen, ob alle Operatoren, Variablen und Prädikaten gleich sind.

Eine weiter Schwierigkeit tritt zutage, wenn die Beweisregel nicht auf die oberste Ebene eines logischen Ausdrucks angewandt wird. Das kann z. B. bei der And-Elimination (vgl. Abbildung 5) der Fall sein.

$$\frac{\phi \wedge \psi}{\phi} \wedge \mathrm{e}_1$$

Abbildung 5: And-Elimination (Huth & Ryan, 2004)

Dazu wurde ein Algorithmus entwickelt, der

1. prüft, ob der oberste Knoten ein FormulaNode ist
2. ist er das nicht, muss es ein UnaryOperatorNode sein. Dieser wird auf Gleichheit überprüft und die check-Methode wird auf den Kindknoten des UnaryOperatorNode rekursiv angewandt
3. handelt es sich um einen FromulaNode und ist der Operator AND, dann muss nur der linke Operand dem Ergebnis-Ausdruck entsprechen.
4. handelt es sihc um einen FormulaNode, aber der Operator ist nicht AND, dann müssen zwei Fälle untersucht werden
    a. im linken Operanden wurde die Umformung vollzogen: dann muss auf den linken Operanden rekursiv check ausgeführt werden und der rechte Operand muss auf Gleichheit überprüft werden
    b. wenn die Umformung im rechten Operanden vollzogen wurde, dann muss auf den rechten Operanden rekursiv check ausgeführt werden und der linke Operand muss auf Gleichheit überprüft werden

Bei einigen Beweisregeln müssen zwei Ausdrücke verglichen werden, die gleich sein sollen, bis auf eine Variable, die immer durch eine andere ersetzt sein muss. Ein Beispiel dafür ist die Forall-Elimination in Abbildung 6.

$$\frac{\forall x \phi}{\phi[t/x]} \forall x\, e$$

Abbildung 6: Forall-Elimination (Huth & Ryan, 2004)

Im unteren Ausdruck muss in Φ die Variable x mit t ersetzt werden. Hierbei wird vorgegangen, wie bei dem Vergleich auf Gleichheit, außer bei den VariableNodes, wird bei der Variablen x geprüft, ob ihr gegenüber immer t steht.

Der in Abbildung 7 aufgeführte Beweis hat einen Fehler. Und zwar ist es nicht erlaubt, dass in einer Box deklarierte Variablen, diese verlassen. Das geschieht jedoch in Zeile 7.

1		$\exists x\, P(x)$	premise
2		$\forall x\, (P(x) \to Q(x))$	premise
3	x_0		
4	x_0	$P(x_0)$	assumption
5		$P(x_0) \to Q(x_0)$	$\forall x$ e 2
6		$Q(x_0)$	$\to$e 5, 4
7		$Q(x_0)$	$\exists x$ e 1, 4–6
8		$\forall y\, Q(y)$	$\forall y$ i 3–7

Abbildung 7: Fehlerhafter Beweis (Huth & Ryan, 2004)

Um das zu überprüfen wurde die Methode variableNotInFormula entwickelt. Diese Methode durchläuft rekursiv einen Ausdruck und prüft für jeden VariableNode, ob dieser nicht die entsprechende Variable ist.

Diese Überprüfung wird in ExistsEliRule angewandt.

3.4 Schnittstelle zum Frontend

Für die Schnittstelle zum Frontend wird Spring Boot verwendet. Es wird ein Controller erstellt, der PredicateController. Dieser greift auf die Klasse PredicateService zu. PredicateService verarbeitet den Eingabestring, indem er geparst wird, der Syntaxbaum wird mit dem predicateProofBaseVisitorImpl verarbeitet und dann das Ergebnis in der Funktion CheckProof.checkProof ausgewertet. Die Ausgabe wird zurück ans Frontend geliefert.

Zur Schnittstellendefinition wird Swagger-Ui verwendet (vgl. Abbildung 8).

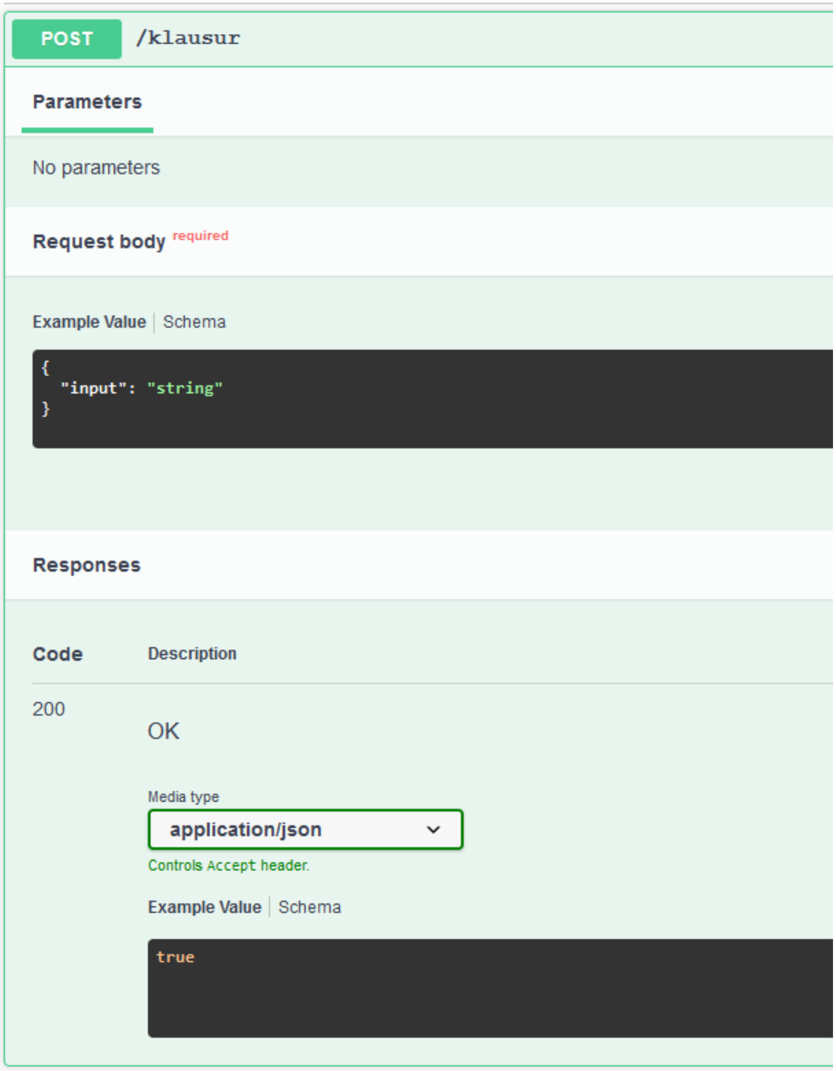


Abbildung 8: Schnittstellendefinition

3.5 Tests

Das System wurde mit den Beweisen aus (Huth & Ryan, 2004) getestet.

Im Anhang befindet sich eine Liste mit den Tests.

4 Zusammenfassung und Diskussion

Können Ranges angegeben, die kein Block sind? Im Buch gab es kein Beispiel.

Equals-Methoden testen

Außer And-Elimination Methoden testen, wo die Umformung nicht auf der obersten Ebene geschehen kann.

Es handelt sich nicht um eine abgeschlossen entwickeltes und getestetes Produkt, sondern um eine Forschungsarbeit.

Das Ergebnis zeigt auf jeden Fall, dass ein entsprechendes Produkt möglich ist. Es wurde auch bereits ein Umfangreicher Lösungsansatz entwickelt, der kleine Schwierigkeiten berücksichtigt.

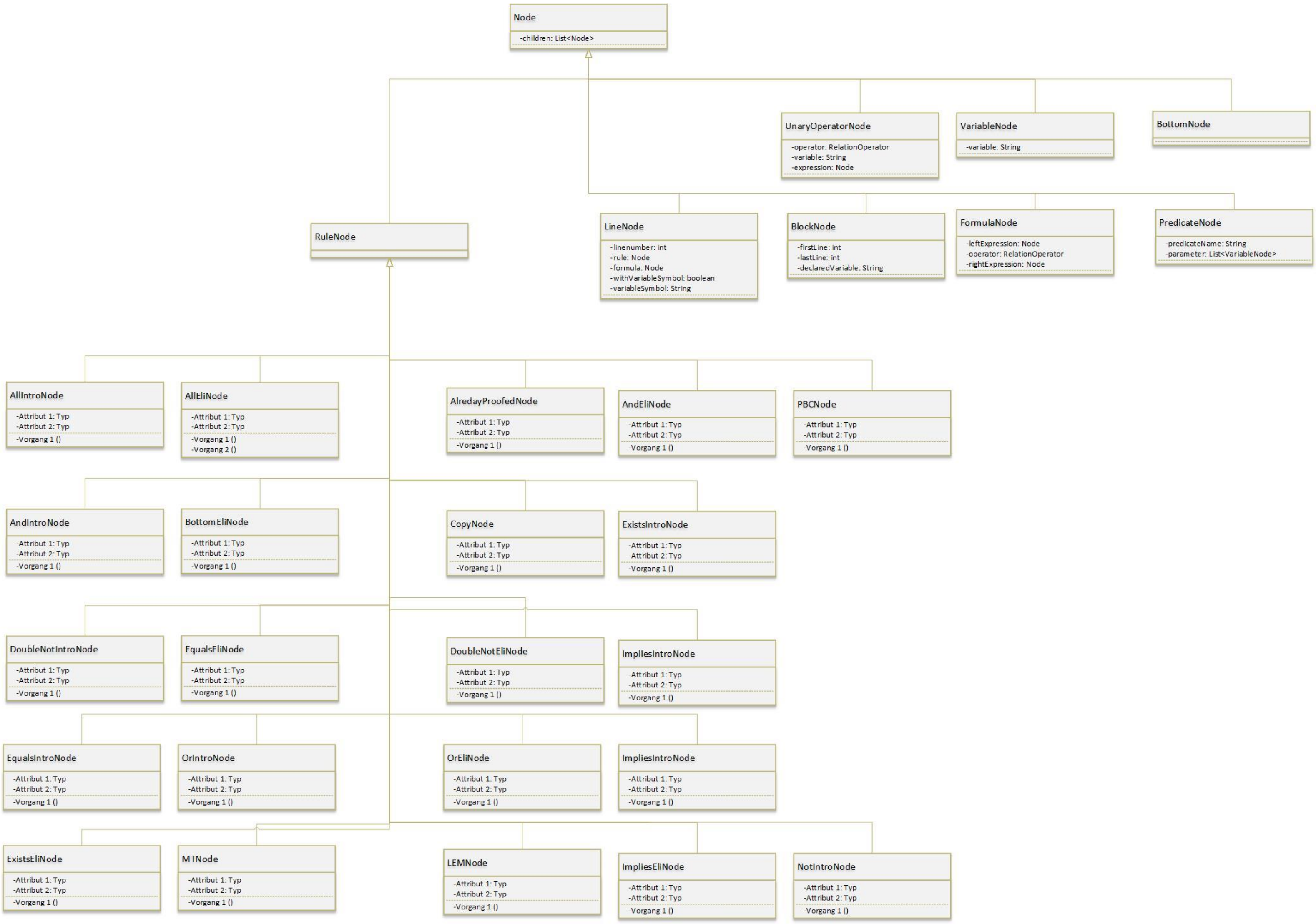
5 Literaturverzeichnis

Huth, M., & Ryan, M. (2004). *Logic in Computer Science.* Cambridge: CAMBRIDGE UNIVERSITY PRESS.

6 Abbildungsverzeichnis

7 Anhang

7.1 Node Datenstruktur



Exemplarisches Klassendiagramm: bei den RuleNodes sind Attribute und Methoden nicht beschrieben.

7.2 Tests

Alle Beispiele sind aus (Huth & Ryan, 2004) entnommen.

Quelle	Eingabe	Eingabe	Soll-Ausgabe
Seite 7	\ll 1 p \wedge q \mid premise \\ 2 r \mid premise \\ 3 q \mid \wedge e2 1 \\ 4 q \wedge r \mid \wedge i 3,2 \\ \gg	1 $p \wedge q$ premise 2 r premise 3 q $\wedge e_2$ 1 4 $q \wedge r$ $\wedge i$ 3, 2	True

Seite 8 Example 1.5	\ll 1 p \mid premise \\ 2 \neg \neg (q \wedge r) \mid premise \\ 3 \neg \neg p \mid \neg \neg i 1 \\ 4 q \wedge r \mid \neg \neg e 2 \\ 5 r \mid \wedge e2 4 \\ 6 \neg \neg p \wedge r \mid \wedge i 3 , 5 \\ \gg	1 p premise 2 $\neg\neg(q \wedge r)$ premise 3 $\neg\neg p$ $\neg\neg$i 1 4 $q \wedge r$ $\neg\neg$e 2 5 r $\wedge e_2$ 4 6 $\neg\neg p \wedge r$ $\wedge$i 3, 5	True
Seite 8 Example 1.6	\ll 1 (p \wedge q) \wedge r \mid premise \\ 2 s \wedge t \mid premise \\ 3 p \wedge q \mid \wedge e1 1 \\ 4 q \mid \wedge e2 3 \\ 5 s \mid \wedge e1 2 \\ 6 q \wedge s \mid \wedge i 4,5 \\ \gg	1 $(p \wedge q) \wedge r$ premise 2 $s \wedge t$ premise 3 $p \wedge q$ $\wedge e_1$ 1 4 q $\wedge e_2$ 3 5 s $\wedge e_1$ 2 6 $q \wedge s$ $\wedge$i 4, 5	True
Seite 9	\ll 1 \neg p \wedge q \mid premise \\ 2 \neg p \wedge q -> r \vee \neg p \mid premise \\ 3 r \vee \neg p \mid -> e 2,1 \\ \gg	1 $\neg p \wedge q$ premise 2 $\neg p \wedge q \to r \vee \neg p$ premise 3 $r \vee \neg p$ $\to$e 2, 1	True
Seite 10	\ll 1 p -> (q -> r) \mid premise \\ 2 p -> q \mid premise \\ 3 p \mid premise \\ 4 q -> r \mid -> e 1,3 \\ 5 q \mid -> e 2,3 \\ 6 r \mid -> e 4,5 \\ \gg	1 $p \to (q \to r)$ premise 2 $p \to q$ premise 3 p premise 4 $q \to r$ $\to$e 1, 3 5 q $\to$e 2, 3 6 r $\to$e 4, 5	True
Seite 10 Example 1.7	\ll 1 p -> (q -> r) \mid premise \\ 2 p \mid premise \\ 3 \neg r \mid premise \\ 4 q -> r \mid -> e 1,2 \\ 5 \neg q \mid MT 4,3 \\ \gg	1 $p \to (q \to r)$ premise 2 p premise 3 $\neg r$ premise 4 $q \to r$ $\to$e 1, 2 5 $\neg q$ MT 4, 3	True
Seite 11 Example 1.8	\ll 1 \neg p -> q \mid premise \\ 2 \neg q \mid premise \\ 3 \neg \neg p \mid MT 1,2 \\ 4 p \mid \neg \neg e 3 \\ \gg	1 $\neg p \to q$ premise 2 $\neg q$ premise 3 $\neg\neg p$ MT 1, 2 4 p $\neg\neg$e 3	True
Seite 11 Example 1.8	\ll 1 p -> \neg q \mid premise \\ 2 q \mid premise \\ 3 \neg \neg q \mid \neg \neg i 2 \\ 4 \neg p \mid MT 1,3 \\ \gg	1 $p \to \neg q$ premise 2 q premise 3 $\neg\neg q$ $\neg\neg$i 2 4 $\neg p$ MT 1, 3	True
Seite 11	\ll 1 p -> q \mid premise \\ \ll 2 \neg q \mid ass \\ 3 \neg p \mid MT 1,2 \\ \gg	1 $p \to q$ premise 2 $\neg q$ assumption 3 $\neg p$ MT 1, 2 4 $\neg q \to \neg p$ $\to$i 2–3	True

	4 \neg q -> \neg p \mid -> i 2-3 \\ \gg		
Seite 13 Example 1.9	\ll 1 \neg q -> \neg p \mid premise \\ \ll 2 p \mid assumption \\ 3 \neg \neg p \mid \neg \neg i 2 \\ 4 \neg \neg q \mid MT 1,3 \\ \gg 5 p -> \neg \neg q \mid -> i 2-4 \\ \gg	1 $\neg q \to \neg p$ premise 2 p assumption 3 $\neg\neg p$ $\neg\neg$i 2 4 $\neg\neg q$ MT 1, 3 5 $p \to \neg\neg q$ $\to$i 2–4	True
Seite 13 Example 1.11	\ll \ll 1 q -> r \mid assumption \\ \ll 2 \neg q -> \neg p \mid ass \\ \ll 3 p \mid ass \\ 4 \neg \neg p \mid \neg \neg i 3 \\ 5 \neg \neg q \mid MT 2,4 \\ 6 q \mid \neg \neg e 5 \\ 7 r \mid -> e 1,6 \\ \gg 8 p -> r \mid -> i 3-7 \\ \gg 9 (\neg q -> \neg p) -> (p -> r) \mid -> i 2-8 \\ \gg 10 (q -> r) -> ((\neg q -> \neg p) -> (p-> r)) \mid -> i 1-9 \\ \gg	1 $q \to r$ assumption 2 $\neg q \to \neg p$ assumption 3 p assumption 4 $\neg\neg p$ $\neg\neg$i 3 5 $\neg\neg q$ MT 2, 4 6 q $\neg\neg$e 5 7 r $\to$e 1, 6 8 $p \to r$ $\to$i 3–7 9 $(\neg q \to \neg p) \to (p \to r)$ $\to$i 2–8 10 $(q \to r) \to ((\neg q \to \neg p) \to (p \to r))$ $\to$i 1–9	True
Seite 15 Example 1.13	\ll 1 p \wedge q -> r \mid premise \\ \ll 2 p \mid ass \\ \ll 3 q \mid ass \\ 4 p \wedge q \mid \wedge i 2,3 \\ 5 r \\mid -> e 1,4 \\ \gg 6 q -> r \mid -> i 3-5 \\ \gg 7 p -> (q -> r) \mid -> i 2-6 \\ \gg	1 $p \wedge q \to r$ premise 2 p assumption 3 q assumption 4 $p \wedge q$ $\wedge$i 2, 3 5 r $\to$e 1, 4 6 $q \to r$ $\to$i 3–5 7 $p \to (q \to r)$ $\to$i 2–6	True
Seite 16 Example 1.14	\ll 1 p -> (q -> r) \mid premise \\ \ll 2 p \wedge q \mid ass \\ 3 p \mid \wedge e1 2 \\ 4 q \mid \wedge e2 2 \\ 5 q -> r \mid -> e 1,3 \\ 6 r \mid -> e 5,4 \\ \gg 7 p \wedge q -> r \mid -> i 2-6 \\ \gg	1 $p \to (q \to r)$ premise 2 $p \wedge q$ assumption 3 p $\wedge e_1$ 2 4 q $\wedge e_2$ 2 5 $q \to r$ $\to$e 1, 3 6 r $\to$e 5, 4 7 $p \wedge q \to r$ $\to$i 2–6	True

Seite 16 Example 1.15	\ll 1 p -> q \mid premise \\ \ll 2 p \wedge r \mid ass \\ 3 p \mid \wedge e1 2 \\ 4 r \mid \wedge e2 2 \\ 5 q \mid -> e 1,3 \\ 6 q \wedge r \mid \wedge i 5,4 \\ \gg 7 p \wedge r -> q \wedge r \mid -> i 2-6 \\ \gg	1 $p \to q$ premise 2 $p \wedge r$ assumption 3 p $\wedge e_1$ 2 4 r $\wedge e_2$ 2 5 q $\to e$ 1, 3 6 $q \wedge r$ $\wedge i$ 5, 4 7 $p \wedge r \to q \wedge r$ $\to i$ 2–6	True
Seite 18 Example 1.16	\ll 1 q -> r \mid premise \\ \ll 2 p \vee q \mid ass \\ \ll 3 p \mid ass \\ 4 p \vee r \mid \vee i1 3 \\ \gg \ll 5 q \mid ass \\ 6 r \mid -> e 1,5 \\ 7 p \vee r \mid \vee i2 6 \\ \gg 8 p \vee r \mid \vee e 2,3-4,5-7 \\ \gg 9 p \vee q -> p \vee r \mid -> i 2-8 \\ \gg	1 $q \to r$ premise 2 $p \vee q$ assumption 3 p assumption 4 $p \vee r$ $\vee i_1$ 3 5 q assumption 6 r $\to e$ 1, 5 7 $p \vee r$ $\vee i_2$ 6 8 $p \vee r$ $\vee e$ 2, 3–4, 5–7 9 $p \vee q \to p \vee r$ $\to i$ 2–8	True
Seite 18 Example 1.17	\ll 1 (p \vee q) \vee r \mid premise \\ \ll 2 (p \vee q) \mid ass \\ \ll 3 p \mid ass \\ 4 p \vee (q \vee r) \mid \vee i1 3 \\ \gg \ll 5 q \mid ass \\ 6 q \vee r \mid \vee i1 5 \\ 7 p \vee (q \vee r) \mid \vee i2 6 \\ \gg 8 p \vee (q \vee r) \mid \vee e 2 , 3-4 , 5-7 \\ \gg \ll 9 r \mid ass \\ 10 q \vee r \mid \vee i2 9 \\ 11 p \vee (q \vee r) \mid \vee i2 10 \\ \gg 12 p \vee (q \vee r) \mid \vee e 1,2-8,9-11 \\ \gg	1 $(p \vee q) \vee r$ premise 2 $(p \vee q)$ assumption 3 p assumption 4 $p \vee (q \vee r)$ $\vee i_1$ 3 5 q assumption 6 $q \vee r$ $\vee i_1$ 5 7 $p \vee (q \vee r)$ $\vee i_2$ 6 8 $p \vee (q \vee r)$ $\vee e$ 2, 3–4, 5–7 9 r assumption 10 $q \vee r$ $\vee i_2$ 9 11 $p \vee (q \vee r)$ $\vee i_2$ 10 12 $p \vee (q \vee r)$ $\vee e$ 1, 2–8, 9–11	True
Seite 19 Example 1.18	\ll 1 p \wedge (q \vee r) \mid premise \\ 2 p \mid \wedge e1 1 \\ 3 q \vee r \mid \wedge e2 1 \\ \ll 4 q \mid ass \\ 5 p \wedge q \mid \wedge i 2,4 \\ 6 (p \wedge q) \vee (p \wedge r) \mid \vee i1 5 \\ \gg \ll 7 r \mid ass \\ 8 p \wedge r \mid \wedge i 2,7 \\	1 $p \wedge (q \vee r)$ premise 2 p $\wedge e_1$ 1 3 $q \vee r$ $\wedge e_2$ 1 4 q assumption 5 $p \wedge q$ $\wedge i$ 2, 4 6 $(p \wedge q) \vee (p \wedge r)$ $\vee i_1$ 5 7 r assumption 8 $p \wedge r$ $\wedge i$ 2, 7 9 $(p \wedge q) \vee (p \wedge r)$ $\vee i_2$ 8 10 $(p \wedge q) \vee (p \wedge r)$ $\vee e$ 3, 4–6, 7–9	True

	9 (p \wedge q) \vee (p \wedge r) \mid \vee i2 8 \\ \gg 10 (p \wedge q) \vee (p \wedge r) \mid \vee e 3,4-6,7-9 \\ \gg		
Seite 21 Example 1.20	\ll 1 \neg p \vee q \mid premise \\ \ll 2 \neg p \mid premise \\ \ll 3 p \mid ass \\ 4 \bot \mid \neg e 3,2 \\ 5 q \mid \bot e 4 \\ \gg 6 p -> q \mid -> i 3-5 \\ \gg \ll 7 q \mid premise \\ \ll 8 p \mid ass \\ 9 \bot \mid \neg e 8,2 \\ 10 q \mid \bot e 9 \\ \gg 11 p -> q \mid -> i 8-10 \\ \gg 12 p->q \mid \vee e 1,2-6,7-11 \\ \gg	1 $\neg p \vee q$ 2 $\neg p$ premise | q premise 3 p assumption | p assumption 4 $\bot$ $\neg$e 3, 2 | q copy 2 5 q $\bot$e 4 | $p \to q$ $\to$i 3–4 6 $p \to q$ $\to$i 3–5 7 $p \to q$ $\vee$e 1, 2–6	True
Seite 22 Example 1.21.1	\ll 1 p -> q \mid premise \\ 2 p -> \neg q \mid premise \\ \ll 3 p \mid ass \\ 4 q \mid -> e 1,3 \\ 5 \neg q \mid -> e 2,3 \\ 6 \bot \mid \neg e 4,5 \\ \gg 7 \neg p \mid \neg i 3-6 \\ \gg	1 $p \to q$ premise 2 $p \to \neg q$ premise 3 p assumption 4 q $\to$e 1, 3 5 $\neg q$ $\to$e 2, 3 6 $\bot$ $\neg$e 4, 5 7 $\neg p$ $\neg$i 3–6	True
Seite 22 Example 1.21.2	\ll 1 p -> \neg p \mid premise \\ \ll 2 p \mid ass \\ 3 \neg p \mid -> e 1,2 \\ 4 \bot \mid \neg e 2,3 \\ \gg 5 \neg p \mid \neg i 2-4 \\ \gg	1 $p \to \neg p$ premise 2 p assumption 3 $\neg p$ $\to$e 1, 2 4 $\bot$ $\neg$e 2, 3 5 $\neg p$ $\neg$i 2–4	True
Seite 22 Example 1.22	\ll 1 p -> (q -> r) \mid premise \\ 2 p \mid premise \\ 3 \neg r \mid premise \\ \ll 4 q \mid ass \\ 5 q -> r \mid -> e 1,2 \\ 6 r \mid -> e 5,4 \\ 7 \bot \mid \neg e 6,3 \\ \gg 8 \neg q \mid \neg i 4-7 \\ \gg	1 $p \to (q \to r)$ premise 2 p premise 3 $\neg r$ premise 4 q assumption 5 $q \to r$ $\to$e 1, 2 6 r $\to$e 5, 4 7 $\bot$ $\neg$e 6, 3 8 $\neg q$ $\neg$i 4–7	True

Seite 23 Example 1.23	\ll 1 p \wedge \neg q -> r \mid premise \\ 2 \neg r \mid premise \\ 3 p \mid premise \\ \ll 4 \neg q \mid ass \\ 5 p \wedge \neg q \mid \wedge i 3,4 \\ 6 r \mid -> e 1,5 \\ 7 \bot \mid \neg e 6,2 \\ \gg 8 \neg \neg q \mid \neg i 4-7 \\ 9 q \mid \neg \neg e 8 \\ \gg	1 $p \wedge \neg q \to r$ premise 2 $\neg r$ premise 3 p premise 4 $\neg q$ assumption 5 $p \wedge \neg q$ $\wedge$i 3, 4 6 r $\to$e 1, 5 7 $\bot$ $\neg$e 6, 2 8 $\neg\neg q$ $\neg$i 4–7 9 q $\neg\neg$e 8	True
Seite 26 Example 1.24	\ll 1 p -> q \mid premise \\ 2 \neg p \vee p \mid LEM \\ \ll 3 \neg p \mid ass \\ 4 \neg p \vee q \mid \vee i1 3 \\ \gg \ll 5 p \mid ass \\ 6 q \mid -> e 1,5 \\ 7 \neg p \vee q \mid \vee i2 6 \\ \gg 8 \neg p \vee q \mid \vee e 2,3-4,5-7 \\ \gg	1 $p \to q$ premise 2 $\neg p \vee p$ LEM 3 $\neg p$ assumption 4 $\neg p \vee q$ $\vee$i$_1$ 3 5 p assumption 6 q $\to$e 1, 5 7 $\neg p \vee q$ $\vee$i$_2$ 6 8 $\neg p \vee q$ $\vee$e 2, 3–4, 5–7	True
Seite 111 1	\ll 1 \forall x (P(x) -> Q(x)) \mid premise \\ 2 \forall x P(x) \mid premise \\ \ll 3 y \parallel P(y) -> Q(y) \mid \forall x e 1 \\ 4 P(y) \mid \forall x e 2 \\ 5 Q(y) \mid -> e 3,4 \\ \gg 6 \forall x Q(x) \mid \forall x i 3-5 \\ \gg	1 $\forall x\,(P(x) \to Q(x))$ premise 2 $\forall x\, P(x)$ premise 3 x_0 $P(x_0) \to Q(x_0)$ $\forall x$ e 1 4 $P(x_0)$ $\forall x$ e 2 5 $Q(x_0)$ $\to$e 3, 4 6 $\forall x\, Q(x)$ $\forall x$ i 3–5	True
Seite 111 2	\ll 1 P(t) \mid premise \\ 2 \forall x (P(x) -> \neg Q(x)) \mid premise \\ 3 P(t) -> \neg Q(t) \mid \forall x e 2 \\ 4 \neg Q(t) \mid -> e 3,1 \\ \gg	1 $P(t)$ premise 2 $\forall x\,(P(x) \to \neg Q(x))$ premise 3 $P(t) \to \neg Q(t)$ $\forall x$ e 2 4 $\neg Q(t)$ $\to$e 3, 1	True
Seite 114 1	\ll 1 \forall x (P(x) -> Q(x)) \mid premise \\ 2 \exists x P(x) \mid premise \\ \ll 3 y \parallel P(y) \mid ass \\ 4 P(y) -> Q(y) \mid \forall x e 1 \\ 5 Q(y) \mid -> e 4,3 \\ 6 \exists x Q(x) \mid \exists x i 5 \\ \gg 7 \exists x Q(x) \mid \exists x e 2,3-6 \\ \gg	1 $\forall x\,(P(x) \to Q(x))$ premise 2 $\exists x\, P(x)$ premise 3 x_0 $P(x_0)$ assumption 4 $P(x_0) \to Q(x_0)$ $\forall x$ e 1 5 $Q(x_0)$ $\to$e 4, 3 6 $\exists x\, Q(x)$ $\exists x$ i 5 7 $\exists x\, Q(x)$ $\exists x$ e 2, 3–6	True

Seite 114 2	\ll 1 \forall x (P(x) -> Q(x)) \mid premise \\ 2 \exists P(x) \mid premise \\ \ll 3 y \parallel P(y) \mid ass \\ 4 P(y) -> Q(y) \mid \forall x e 1 \\\ 5 Q(y) \mid -> e 4,3 \\ \gg 6 Q(y) \mid \exists x e 2,3-5 \\ 7 \exists x Q(x) \mid \exists x i 6 \\ \gg	1 $\forall x\,(P(x) \to Q(x))$ premise 2 $\exists x\, P(x)$ premise 3 x_0 $P(x_0)$ assumption 4 $P(x_0) \to Q(x_0)$ $\forall x$ e 1 5 $Q(x_0)$ $\to$e 4, 3 6 $Q(x_0)$ $\exists x$ e 2, 3–5 7 $\exists x\, Q(x)$ $\exists x$ i 6	False
Seite 115	\ll 1 \forall x (Q(x) -> R(x)) \mid premise \\ 2 \exists x (P(x) \wedge Q(x)) \mid premise \\ \ll 3 y \parallel P(y) \wedge Q(y) \mid ass \\ 4 Q(y) -> R(y) \mid ass \\ 5 Q(y) \mid \wedge e2 3 \\ 6 R(y) \mid -> e 4,5 \\ 7 P(y) \mid \wedge e1 3 \\ 8 P(y) \wedge R(y) \mid \wedge i 7,6 \\ 9 \exists x (P(x) \wedge R(x)) \mid \exists x i 8 \\ \gg 10 \exists x (P(x) \wedge R(x)) \mid \exists x e 2,3-9 \\ \gg	1 $\forall x\,(Q(x) \to R(x))$ premise 2 $\exists x\,(P(x) \wedge Q(x))$ premise 3 x_0 $P(x_0) \wedge Q(x_0)$ assumption 4 $Q(x_0) \to R(x_0)$ $\forall x$ e 1 5 $Q(x_0)$ $\wedge$e$_2$ 3 6 $R(x_0)$ $\to$e 4, 5 7 $P(x_0)$ $\wedge$e$_1$ 3 8 $P(x_0) \wedge R(x_0)$ $\wedge$i 7, 6 9 $\exists x\,(P(x) \wedge R(x))$ $\exists x$ i 8 10 $\exists x\,(P(x) \wedge R(x))$ $\exists x$ e 2, 3–9	True
Seite 116 1	\ll 1 \exists x P(x) \mid premise \\ 2 \forall x \forall y (P(x) -> Q(y)) \mid premise \\ \ll 3 w \parallel \\ \ll 4 v \parallel P(v) \mid ass \\ 5 \forall y (P(v) -> Q(y)) \mid \forall x e 2 \\ 6 P(v) -> Q(w) \mid \forall y e 5 \\ 7 Q(w) \mid -> e 6,4 \\ \gg 8 Q(w) \mid \exists x e 1, 4-7 \\ \gg 9 \forall y Q(y) \mid \forall y i 3-8 \\ \gg	1 $\exists x\, P(x)$ premise 2 $\forall x \forall y\,(P(x) \to Q(y))$ premise 3 y_0 4 x_0 $P(x_0)$ assumption 5 $\forall y\,(P(x_0) \to Q(y))$ $\forall x$ e 2 6 $P(x_0) \to Q(y_0)$ $\forall y$ e 5 7 $Q(y_0)$ $\to$e 6, 4 8 $Q(y_0)$ $\exists x$ e 1, 4–7 9 $\forall y\, Q(y)$ $\forall y$ i 3–8	True
Seite 116 2	\ll 1 \exists x P(x) \mid premise \\ 2 \forall x (P(x) -> Q(x)) \mid premise \\ \ll 3 v \parallel \\ \ll 4 v \mid P(v) \mid ass \\ 5 P(v) -> Q(v) \mid \forall x e 2 \\ 6 Q(v) \mid -> e 5,4 \\ \gg 7 Q(v) \mid \exists x e 1,4-6 \\ \gg 8 \forall y Q(y) \mid \forall y i 3-7 \\ \gg	1 $\exists x\, P(x)$ premise 2 $\forall x\,(P(x) \to Q(x))$ premise 3 x_0 4 x_0 $P(x_0)$ assumption 5 $P(x_0) \to Q(x_0)$ $\forall x$ e 2 6 $Q(x_0)$ $\to$e 5, 4 7 $Q(x_0)$ $\exists x$ e 1, 4–6 8 $\forall y\, Q(y)$ $\forall y$ i 3–7	False