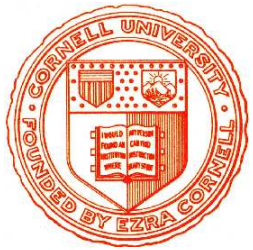
Automatisierte Logik und Programmierung

Einheit 3



Formale Logik (Teil 2)

1. Formalisierung in Aussagenlogik
2. Evidenz für logische Aussagen
3. Evidenzkonstruktion mit Refinement Logik
4. Formale Prädikatenlogik
5. Metamathematik der Refinement Logik

Das übliche Verständnis des Begriffs "Logik"

- **Ermöglicht Formulierung universeller Zusammenhänge**

  ... und ihre Anwendung auf Individuen

  "*Jeder Mensch ist sterblich.*
  Sokrates ist ein Mensch.
  Also ist Sokrates sterblich"

  $$((\forall x)(Human(x) \Rightarrow Mortal(x)) \wedge Human(sokrates)) \Rightarrow Mortal(sokrates)$$

- **Unterstützt unterspezifizierte Aussagen und Funktionen**

  "*Studierende, die mindestens 120 Leistungspunkte erworben haben, können ein Thema für die Bachelorarbeit bekommen*"

  $$(\forall st)\ (lp(s) \geq 120 \Rightarrow (\exists t)\ (BA(t) \wedge Bekommt(s,t)))$$

- **Erweiterung der Aussagenlogik**
  - Syntax wird ergänzt um Variablen, Funktionen, und Quantoren
  - Neue Konzepte: Bindungsbereich, Variablenvorkommen und Substitution

Syntax der Prädikatenlogik

- **Erlaubte Symbole**
  - Variablen $x, y, z, x_0, y_0,, \ldots$
  - Funktionssymbole $f, g, h, a, b, c, f_0, g_0,, \ldots$ (mit Stelligkeit, a, b, c oft nullstellig)
  - Prädikatssymbole $P, Q, R, P_o, Q_0, R_0,, \ldots$ (mit Stelligkeit)
  - Logische Symbole f, $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\forall$, $\exists$ und Klammern
- **Terme: Syntax für individuelle Objekte**
  - Variablen und nullstellige Funktionen (Konstante) sind (atomare) Terme
  - Sind $t_1, .., t_n$ Terme und f n-stellige Funktion, dann ist $f(t_1, \ldots, t_n)$ Term
- **Formeln: Syntax für Aussagen**
  - f und nullstellige Prädikate (Aussagenvariablen) sind (atomare) Formeln
  - $P(t_1, \ldots, t_n)$ ist (atomare) Formel ($t_1, .., t_n$ Terme, P n-stelliges Prädikat)
  - Sind A und B Formeln, dann auch $\neg A$, $(A \Rightarrow B)$, $(A \wedge B)$, $(A \vee B)$
  - Ist B Formel und x eine Variable, dann sind $(\forall x)B$ und $(\exists x)B$ Formeln

    Bindungsbereich des Quantors (Scope) ist die kürzeste Formel, die auf den Quantor folgt

Später: alternative Notationen $\forall x.\, B$ und $\exists x.\, B$ und Konventionen, Klammern zu sparen

Semantik der Prädikatenlogik

- **Evidenz für Gültigkeit von Formeln**
  - Formuliert als Terme in erweiterter λ-Notation (Einheit 5)
  - Konstruktion von Evidenz folgt induktivem Aufbau der Syntax
- **Evidenz für atomare Formeln**
  - f hat keine Evidenz $[\mathsf{f}] = \{\}$
  - $A = P(t_1, ..t_n)$ steht für unbekannte Aussagen $[A]$ **unspezifiziert**
- **"Aussagenlogische" Evidenzkonstruktion wie zuvor**
  - Implikation $[A \Rightarrow B] = [A] \to [B]$
  - Konjunktion $[A \wedge B] = [A] \times [B]$
  - Disjunktion $[A \vee B] = [A] + [B]$
  - Negation $[\neg A] = [A] \to \{\}$

Wie konstruiert man Evidenz für quantifizierte Formeln?

Evidenz für Universelle Quantifikation

- **$(\forall x)B$: "Für alle x gilt B"** $\quad [(\forall x)B] = x : \mathbb{U} \to [B]$
  - Für jede Instanz von x muß eine Evidenz b für B konstruiert werden
  - Evidenz für $(\forall x)B$ muß Funktion f sein mit $f(x) : [B]$ für alle x
  - Eingabe x für f stammt aus einem Universum von Objekten $\mathbb{U}$
  - Ausgabetyp $[B]$ von f kann von konkretem Eingabewert $x : \mathbb{U}$ abhängen
    z.B. $B = (P\,a \Rightarrow P\,x)$ hat genau dann Evidenz, wenn x mit a instantiiert
  - Typ der Evidenzen für $(\forall x)B$ ist ein "abhängiger" Funktionenraum
- **Konkrete Evidenz für $(\forall x)(P\,x \Rightarrow P\,x)$**
  - Evidenz ist Funktion $f : (x{:}\mathbb{U} \to ([P\,x] \to [P\,x]))$, wobei für alle x gilt
    $f(x) = g_x : [P\,x] \to [P\,x]$ und $q_x(p) : [P\,x]$, falls $p : [P\,x]$
  - Einfachste Lösung ist $g_x(p) = p$, also $\quad f = \lambda x.\,(\lambda p.\,p)$
- **Konkrete Evidenz für $((\forall x)P\,x) \Rightarrow P\,a$**
  - Evidenz ist Funktion $f : (x{:}\mathbb{U} \to [P\,x]) \to [P\,a]))$ mit $f(h) = x_h : [P\,a]$
    für alle $h : (x{:}\mathbb{U} \to [P\,x])$.
  - Einfachste Lösung ist Anwendung von h auf Konstante a $\quad f = \lambda h.\,h(a)$

Evidenz für Existentielle Quantifikation

- **$(\exists x)B$: "Es gibt ein x, für das B gilt"** $[(\exists x)B] = x : \mathbb{U} \times [B]$
  - Um Evidenz für $(\exists x)B$ zu konstruieren, braucht man $b : [B]$ für ein $x : \mathbb{U}$
  - Formel B kann von Wahl des konkreten Wertes für x abhängen
  - Typ der Evidenzen für $(\exists x)B$ ist ein "abhängiger" Produktraum
- **Konkrete Evidenz für $P\,a \Rightarrow ((\exists x)P\,x)$**
  - Evidenz ist Funktion $f : (p{:}[P\,a] \to (x : \mathbb{U} \times [P\,x]))$ mit $f(p) = (x, p')$ für alle $p : [P\,a]$, wobei $x : \mathbb{U}$ und p' Evidenz für $P\,x$
  - Einfachste Lösung ist $x = a$ and $p' = p$ $\qquad f = \lambda p.\,(a, p)$
- **Konkrete Evidenz für $((\exists x)P\,x) \Rightarrow ((\exists y)Py)$**
  - Evidenz ist Funktion f mit $f(z) = (y, p')$ für alle $z : (x : \mathbb{U} \times [P\,x])$, wobei $y : \mathbb{U}$ und p' Evidenz für $P\,y$
  - z muß ein Paar (a, p) mit $a : \mathbb{U}$ und $p : [P\,a]$
  - Einfachste Lösung: $x = a = z.1$ and $p' = p = z.2$, also $f = \lambda z.\,(z.1, z.2)$
  - Wegen $z = (z.1, z.2)$ kann Evidenz vereinfacht werden zu $\qquad f = \lambda z.\,z$

Evidenz für komplexere Formeln

- $((\forall x)(P\,x \wedge Q\,x)) \Rightarrow ((\forall x)P\,x \wedge (\forall x)Q\,x)$
  - Evidenz ist Funktion f so daß für alle $h : (x{:}\mathbb{U}{\to}[P\,x]{\times}[Q\,x])$ gilt
    $f(h) = (g_p, g_q) : (x{:}\mathbb{U}{\to}[P\,x]){\times}(x{:}\mathbb{U}{\to}[Q\,x])$
  - g_p und g_q nehmen ein $x{:}\mathbb{U}$ und erzeugen Elemente von $[P\,x]$ bzw. $[Q\,x]$
  - Einfachste Lösung ist $g_p(x) = h(x).1$ und $g_q(x) = h(x).2$

    $f = \boldsymbol{\lambda h.\,(\lambda x.\,h(x).1,\ \lambda x.\,h(x).2)}$
- **Evidenz für** $\neg((\forall x)\neg(P\,x)) \Rightarrow (\exists x)P\,x$
  - Evidenz ist Funktion f so daß für alle $h : (x{:}\mathbb{U}{\to}([P\,x]{\to}\{\}){\to}\{\}$ gilt
    $f(h) = (x, p)$ mit $p : [P\,x]$
  - Zur Konstruktion von $f(h)$ benötigt man Kenntnisse über P und Objekte x, für die $P\,x$ gilt
  - Es gibt keinen allgemeinen Weg, x oder p aus h zu konstruieren, solange das Prädikatssymbol P unspezifiziert ist

Keine universelle Evidenz

Evidenzsemantik – Zusammenfassung

Aussage A	Evidenztyp $[A]$	Evidenzkonstruktion	Dekompositionsterm
$A \Rightarrow B$	$[A] \to [B]$	$\lambda a.b$	$f(a)$
$A \wedge B$	$[A] \times [B]$	(a, b)	$x.1$, $x.2$
$A \vee B$	$[A] + [B]$	$\texttt{inl}(a)$, $\texttt{inr}(b)$	$\texttt{case}\ x\ \texttt{of}\ \texttt{inl}(a) \to s \mid \texttt{inr}(b) \to t$
$\neg A$	$[A] \to \{\}$	$\lambda a.b$	$f(a)$
f	$\{\}$	–	–
$(\forall x)B$	$x : \mathbb{U} \to [B]$	$\lambda a.b$	$f(a)$
$(\exists x)B$	$x : \mathbb{U} \times [B]$	(a, b)	$x.1$, $x.2$

- **Formeln korrespondieren mit Datentyp ihrer Evidenzen**
  - Es gibt Terme, um Evidenz zu konstruieren oder zu zerlegen
  - Beide sind invers zueinander und ermöglichen "Rechnen" mit Evidenz

(Einheit 5)

- **Evidenzterme bilden eine Programmiersprache**
  - Sprache umfasst Terme der Prädikatenlogik
  - Prädikatenlogik kann nur Programme ohne Schleifen spezifizieren
  - Mehr Ausdruckskraft benötigt Gleichheit, Zahlen, Induktion, ...

Prädikatenlogische Refinement Logik

- **Erweiterung der aussagenlogische Regeln**
  - Regel `axiom` ist auf atomare prädikatenlogische Formeln anwendbar
  - Unveränderte Regeln für $A \Rightarrow B$, $A \wedge B$, $A \vee B$, $\neg A$
  - Konstruktierte Evidenz ändert sich ebenfalls nicht
- **Behandlung von Quantoren**
  - Um $(\forall x) B$ zu zeigen, muß man B für jede Instanz von x zeigen
    Hierzu wählt man $x' : \mathbb{U}$ beliebig aber fest und zeigt B für x' statt x
  - Um $(\exists x) B$ zu zeigen, muß man eine B für eine Instanz von x zeigen
    Hierzu gibt man ein Objekt a an und zeigt B für a statt x
  - Regeln benötigen "syntaktische Instantiierung" von Variablen
- **Formales Konzept: Substitution $B[t/x]$**
  - Ersetzen der Variablen x in Formel B durch Term t
    Unvollständiger Ersatz für Instantiierung, wenn Universum überabzählbar
  - Substitution muß Verständnis von "Für alle" und "es gibt" erhalten
    $(\forall x) P\, x$ und $(\forall y) P\, y$ bedeuten dasselbe
  - Nur ungebundene Variablen dürfen ersetzt werden

Vorkommen von Variablen in Formeln

- **Vorkommen der Variablen x in Formel B, informal**
  - Gebunden: x erscheint im Scope eines Quantors $(\forall x)$ oder $(\exists x)$
  - Frei: x kommt in B vor, ohne gebunden zu sein
  - B heißt geschlossen falls B keine freien Variablen enthält
- **Präzise, induktive Definition**

x	die Variable x kommt frei vor; $y \neq x$ kommt nicht vor
f	die Variable x kommt nicht vor
$f(t_1, ..., t_n)$	freie Vorkommen von x in t_i bleiben frei
$P(t_1, ..., t_n)$	gebundene Vorkommen von x bleiben gebunden.
$\neg A$, $A \Rightarrow B$	freie Vorkommen von x in A, B bleiben frei
$A \wedge B$, $A \vee B$	gebundene Vorkommen von x bleiben gebunden.
$(\forall x)B$	beliebige Vorkommen von x in B werden gebunden
$(\exists x)B$	Vorkommen von $y \neq x$ in B bleiben unverändert

$$\overbrace{\overbrace{(\forall x)\underbrace{(P(x) \wedge Q(x))}_{x \text{ frei}}}^{x \text{ gebunden}} \wedge \underbrace{R(x)}_{x \text{ frei}}}^{x \text{ frei und gebunden}}$$

Substitution $B[t/x]$ formal

Endliche Abbildung σ von Variablen in Terme

– $\boldsymbol{\sigma = [t_1, .., t_n/x_1, .., x_n]} \mathrel{\hat{=}} \sigma(x_1){=}t_1, \ldots, \sigma(x_n){=}t_n$

– $\boldsymbol{A\sigma}$: Anwendung von σ auf den Ausdruck A τ und σ

$\lfloor x \rfloor[t/x]$	$= t$	$\lfloor x \rfloor[t/y]$	$= x$	$(y{\neq}x)$
$\lfloor f(t_1, .., t_n) \rfloor\sigma$	$= f(t_1\sigma, .., t_n\sigma)$	$\lfloor \mathsf{f} \rfloor\sigma$	$= \mathsf{f}$	
$\lfloor P(t_1, .., t_n) \rfloor\sigma$	$= P(t_1\sigma, .., t_n\sigma)$			
$\lfloor \neg A \rfloor\sigma$	$= \neg A\sigma$	$\lfloor A \wedge B \rfloor\sigma$	$= A\sigma \wedge B\sigma$	
$\lfloor A \vee B \rfloor\sigma$	$= A\sigma \vee B\sigma$	$\lfloor A \Rightarrow B \rfloor\sigma$	$= A\sigma \Rightarrow B\sigma$	
$\lfloor (\forall x)B \rfloor[t/x]$	$= (\forall x)B$	$\lfloor (\exists x)B \rfloor[t/x]$	$= (\exists x)B$	
$\lfloor (\forall x)B \rfloor[t/y]$	$= \lfloor (\forall z)B[z/x] \rfloor[t/y]$	$\lfloor (\exists x)B \rfloor[t/y]$	$= \lfloor (\exists z)B[z/x] \rfloor[t/y]$	*
$\lfloor (\forall x)B \rfloor[t/y]$	$= (\forall x) . \lfloor B[t/y] \rfloor$	$\lfloor (\exists x)B \rfloor[t/y]$	$= (\exists x) \lfloor B[t/y] \rfloor$	**

*: *$y{\neq}x$, y frei in B, x frei in t, z neue Variable*

**: *$y{\neq}x$, y nicht frei in B oder x nicht frei in t*

Regeln für Allquantor

- **Allquantor auf rechter Seite einer Sequenz**
  - Um $H \vdash (\forall x)B$ zu zeigen, muß man B für jede Instanz von x zeigen
    Einziger Weg ist generischer Beweis, der nicht von Instanz abhängt
  - Wähle neue Variable x' und beweise $B[x'/x]$

$$\frac{H \vdash (\forall x)B \qquad \mathtt{ev} = \lambda x'.\, b \quad\quad H, x':\mathbb{U} \vdash B[x'/x] \qquad \mathtt{ev} = b}{}\ \texttt{allR}$$

  - Im Teilziel wird generische Evidenz b für $B[x'/x]$ und alle x' konstruiert
  - Evidenz für $(\forall x)B$ muß Funktion $\lambda x'.b$ sein
- **Allquantor auf linker Seite einer Sequenz**
  - Um C unter Annahme $(\forall x)B$ zu zeigen, darf man jede Instanz von x verwenden, also die Annahme $B[t/x]$ für beliebige Terme t ergänzen

$$\frac{H, f:(\forall x)B, H' \vdash C \qquad \mathtt{ev} = c[f(t)/b] \quad\quad H, f:(\forall x)B, b:B[t/x], H' \vdash C \qquad \mathtt{ev} = c}{}\ \texttt{allL}\ t$$

  - Regel nimmt an, daß Evidenz c aus Evidenz $b:B[t/x]$ konstruierbar ist
  - Anwendung von $\lambda b.c$ auf $f(t)$ liefert Evidenz für C im Hauptziel

Anwendung der Allquantorregeln

- **Beweis für $(\forall x)(Px \Rightarrow Px)$**

$\vdash\ (\forall x)(Px \Rightarrow Px)$	ev = $\lambda x.\,(\lambda p.p)$	BY allR
1 $x{:}\mathbb{U} \vdash\ Px \Rightarrow Px$	ev = $\lambda p.p$	BY impliesR
1.1 $x{:}\mathbb{U}, p{:}Px \vdash\ Px$	ev = p	BY axiom

- **Beweis für $((\forall x)Px) \Rightarrow Pa$**

$\vdash ((\forall x)Px) \Rightarrow Pa$	ev = $\lambda f.\,f(a)$	BY impliesR
1 $f{:}(\forall x)Px \vdash Pa$	ev = $p[f(a)/p]$	BY allL a
1.1 $f{:}(\forall x)Px, p{:}Pa \vdash Pa$	ev = p	BY axiom

- **Beweis für $((\forall x)Px) \Rightarrow (Pa \wedge Pb)$**

$\vdash ((\forall x)Px) \Rightarrow (Pa \wedge Pb)$	ev = $\lambda f.(f(a), f(b))$	BY impliesR
1 $f{:}(\forall x)Px \vdash Pa \wedge Pb$	ev = $(f(a), f(b))$	BY allL a
1.1 $f{:}(\forall x)Px, p_a{:}Pa \vdash Pa \wedge Pb$	ev = $(p_a, f(b))$	BY allL b
1.1.1 $f{:}(\forall x)Px, p_a{:}Pa, p_b{:}Pb \vdash Pa \wedge Pb$	ev = (p_a, p_b)	BY andR
1.1.1.1 $f{:}(\forall x)Px, p_a{:}Pa, p_b{:}Pb \vdash Pa$	ev = p_a	BY axiom
1.1.1.2 $f{:}(\forall x)Px, p_a{:}Pa, p_b{:}Pb \vdash Pb$	ev = p_b	BY axiom

Regeln für Existenzquantor

- **Existenzquantor auf rechter Seite einer Sequenz**
  - Um $H \vdash (\exists x)B$ zu zeigen, muß $B[t/x]$ für einen Term t gezeigt werden

$$\frac{H \vdash (\exists x)B \qquad \mathtt{ev} = (t, b)}{H \vdash B[t/x] \qquad \mathtt{ev} = b}\ \mathtt{exR}\ t$$

  - Regel setzt Term t und Evidenz $b : B[t/x]$ zur Evidenz (t, b) zusammen
- **Existenzquantor auf linker Seite einer Sequenz**
  - Um C unter Annahme $(\exists x)B$ zu beweisen, muß man C unter Annahme B für eine beliebige Instanz von x, also generisch, zeigen können
  - Wähle neue Variable x' und verwende Annahme $B[x'/x]$

$$\frac{H, z{:}(\exists x)B, H' \vdash C \qquad \mathtt{ev} = c[z.1, z.2/x', b]}{H, x'{:}\mathbb{U}, b{:}B[x'/x], H' \vdash C \qquad \mathtt{ev} = c}\ \mathtt{exL}$$

  - Label z für $(\exists x)B$ entspricht Paar aus Variablen x' und Label $b{:}B$
  - Evidenz c hängt im Teilziel von x' und b ab
  - Im Hauptziel muß x' durch $z.1$ und b durch $z.2$ ersetzt werden

Anwendung der Existenzquantorregeln

- **Beweis für $Pa \Rightarrow ((\exists x)Px)$**

$\vdash\ Pa \Rightarrow ((\exists x)Px)$	ev = $\lambda p.(a, p)$	BY impliesR
1 $p{:}Pa \vdash\ (\exists x)Px$	ev = (a, p)	BY exR a
1.1 $p{:}Pa \vdash\ Pa$	ev = p	BY axiom

- **Beweis für $((\exists x)Px) \Rightarrow ((\exists y)Py)$**

$\vdash\ ((\exists x)Px) \Rightarrow ((\exists y)Py)$	ev = $\lambda z.\,(z.1, z.2)$	BY impliesR
1 $z{:}(\exists x)Px \vdash\ (\exists y)Py$	ev = $(z.1, z.2)$	BY exL
1.1 $x{:}\mathbb{U}, p{:}Px \vdash\ (\exists y)Py$	ev = (x, p)	BY exR x
1.1.1 $x{:}\mathbb{U}, p{:}Px \vdash\ Px$	ev = p	BY axiom

  - Evidenz $(z.1, z.2)$ kann zu z vereinfacht werden
  - Reihenfolge der Regelanwendungen wichtig für erfolgreichen Beweis

$\vdash\ ((\exists x)Px) \Rightarrow ((\exists y)Py)$	BY impliesR
1 $z{:}(\exists x)Px \vdash\ (\exists y)Py$	BY exR x
1.1 $z{:}(\exists x)Px \vdash\ Px$	BY exL
1.1.1 $x'{:}\mathbb{U}, p{:}Px' \vdash\ Px$	BY ???

Ein komplexerer Beweis

$\vdash ((\forall x)(Px \wedge Qx)) \Rightarrow ((\forall x)Px \wedge (\forall x)Qx)$	BY `impliesR`
1. $f{:}(\forall x)(Px \wedge Qx) \vdash ((\forall x)Px \wedge (\forall x)Qx)$	BY `andR`
1.1. $f{:}(\forall x)(Px \wedge Qx) \vdash (\forall x)Px$	BY `allR`
1.1.1. $x{:}\mathbb{U}, f{:}(\forall x)(Px \wedge Qx) \vdash Px$	BY `allL` x
1.1.1.1. $x{:}\mathbb{U}, f{:}(\forall x)(Px \wedge Qx), z{:}(Px \wedge Qx) \vdash Px$	BY `andL`
1.1.1.1.1. $x{:}\mathbb{U}, f{:}(\forall x)(Px \wedge Qx), p{:}Px, q{:}Qx \vdash Px$	BY `axiom`
1.2. $f{:}(\forall x)(Px \wedge Qx) \vdash (\forall x)Qx$	BY `allR`
1.2.1. $x{:}\mathbb{U}, f{:}(\forall x)(Px \wedge Qx) \vdash Qx$	BY `allL` x
1.2.1.1. $x{:}\mathbb{U}, f{:}(\forall x)(Px \wedge Qx), z{:}(Px \wedge Qx) \vdash Qx$	BY `andL`
1.2.1.1.1. $x{:}\mathbb{U}, f{:}(\forall x)(Px \wedge Qx), p{:}Px, q{:}Qx \vdash Qx$	BY `axiom`

Konstruierte Evidenz ist $\lambda f.\,(\lambda x.(f\,x).1, \lambda x.(f\,x).2)$

Refinement Logik – Zusammenfassung

Links			Rechts		
$H, f{:}A \Rightarrow B, H' \vdash C$	ev=$c[f(a)/b]$	impliesL	$H \vdash A \Rightarrow B$	ev=$\lambda a.b$	impliesR
$H, f{:}A \Rightarrow B, H' \vdash A$	ev=a		$H, a{:}A \vdash B$	ev=b	
$H, b{:}B, H' \vdash C$	ev=c				
$H, x{:}A \wedge B, H' \vdash C$	ev=$c[x.1, x.2/a, b]$	andL	$H \vdash A \wedge B$	ev=(a, b)	andR
$H, a{:}A, b{:}B, H' \vdash C$	ev=c		$H \vdash A$	ev=a	
			$H \vdash B$	ev=b	
$H, x{:}A \vee B, H' \vdash C$	ev=`case` x `of inl(`a`)`$\rightarrow c_1$	orL	$H \vdash A \vee B$	ev=`inl(`a`)`	orR1
$H, a{:}A, H' \vdash C$	ev=c_1 `| inr(`b`)`$\rightarrow c_2$		$H \vdash A$	ev=a	
$H, b{:}B, H' \vdash C$	ev=c_2		$H \vdash A \vee B$	ev=`inr(`b`)`	orR2
			$H \vdash B$	ev=b	
$H, f{:}\neg A, H' \vdash C$	ev=`any(`$f(a)$`)`	notL	$H \vdash \neg A$	ev=$\lambda a.b$	notR
$H, f{:}\neg A, H' \vdash A$	ev=a		$H, a{:}A \vdash \mathsf{f}$	ev=b	
			$H, a{:}A, H' \vdash A$	ev=a	axiom
$H, f{:}(\forall x)B, H' \vdash C$	ev=$c[f(t)/b]$	allL t	$H \vdash (\forall x)B$	ev=$\lambda x'.\, b$	allR
$H, f{:}(\forall x)B, b{:}B[t/x], H' \vdash C$	ev=c		$H, x'{:}\mathbb{U} \vdash B[x'/x]$	ev=b	
$H, z{:}(\exists x)B, H' \vdash C$	ev=$c[z.1, z.2/x', b]$	exL	$H \vdash (\exists x)B$	ev=(t, b)	exR t
$H, x'{:}\mathbb{U}, b{:}B[x'/x], H' \vdash C$	ev=c		$H \vdash B[t/x]$	ev=b	

t ist ein beliebiger Term, x' eine neue Variable

Sinnvolle Zusatzregeln

- **Einfügen von Zwischenbehauptungen**
  - C ist gültig, wenn C aus der Annahme A folgt und A gültig ist

$$\frac{H \vdash C \quad \mathtt{ev} = (\lambda x.c)(a) \qquad H \vdash A \quad \mathtt{ev} = a \qquad H, x{:}A \vdash C \quad \mathtt{ev} = c}{\mathtt{cut}\ A}$$

  - A kann eine beliebige "Schnitt"-Formel sein
  - Beweise werden signifikant kürzer, wenn A mehrfach benutzt wird
  - Evidenz $(\lambda x.c)(a)$ wird nicht evaluiert (Gefahr der Termaufblähung)
- **Ausdünnen von Annahmen**
  - Hypothesen, die nicht gebraucht werden, können entfernt werden

$$\frac{H, a{:}A, H' \vdash C \quad \mathtt{ev} = c}{H, H' \vdash C \quad \mathtt{ev} = c}\ \mathtt{thin}\ A$$

  - Sinnvoll, um Hypothesenliste übersichtlich zu halten
  - Evidenz c hängt im Hauptziel nicht vom Label a ab

Aussagen über den logischen Kalkül

- **Wichtig für Analyse der Eigenschaften des Kalküls**
  - Ist Refinement Logik korrekt? (sind beweisene Formeln gültig?
  - Ist Refinement Logik vollständig? (sind gültige Formeln beweisbar?)
  - Welche Formeln sind entscheidbar?
  - Was ist die Komplexität von Beweissuche?
- **Hilfreich für Implementierung und Automatisierung**
  - Beschreibung der Struktur der Grundelemente des Kalküls
  - Daten- und Zugriffsstrukturen für Formeln, Beweise, Evidenz, ...
  - Algorithmen zur Anwendung von Regeln und Evidenzkonstruktion
  - Benutzerdefinierbare ("konservative") Erweiterung von Objektsprache und Regelsystem durch Definitionen und Beweisstrategien

Unterschied Objekt- / Metasprache

Präsentation von Kalkülen hat zwei Sprachebenen

- **Objektsprache:**
  - Sprache des Kalküls, in dem formalisiert wird
  - Formale Sprache mit präzise definierter Syntax
  - Konkretes Element z.B. $(\exists x)(P\,x) \vee Q\,x) \Rightarrow (\neg((\forall x)(\neg P\,x \wedge \neg Q\,x)))$
- **Metasprache:**
  - Sprache, um Aussagen über den Kalkül zu machen
    - Beschreibung von Syntax, Semantik, Eigenschaften des Kalküls
  - Natürliche, oft stark schematisierte Sprache
  - Enthält Objektsprache, angereichert um syntaktische Metavariablen
  - Konkretes Element z.B. *aus* $(\exists x)\,(A \vee B)$ *folgt* $\neg((\forall x)\,(\neg A \wedge \neg B))$
- **Unterscheidung zuweilen durch Fonts / Farben**
  - Ansonsten aus Kontext eindeutig erkennbar

Refinement Logik als formaler Kalkül (I)

Metasprachliche Präzisierung der verwendeten Konzepte

- **Kalkül verwendet Objekt-Logik / Evidenz als Parameter**
  - Objektsprache ist (bisher) Sprache der Prädikatenlogik
  - Evidenz formuliert als Terme in erweiterter λ-Notation
  - Semantik der Logik erklärt, wann Terme Evidenz für Formeln sind
- **Sequenz (Ziel)** $\boldsymbol{H \vdash C}$
  - $H = x_1{:}A_1, .., x_n{:}A_n$ Hypothesenliste (x_i verschieden), C Konklusionsformel
  - $x_i : A_i$ Deklaration "x_i ist Evidenz für A_i" (x Term-Variable, A Formel)
  - Initialsequenz: Sequenz $\vdash C$ ohne Hypothesen
- **Evidenzterm für** $\boldsymbol{x_1{:}A_1, .., x_n{:}A_n \vdash C}$
  - Term e mit freien Variablen aus den x_i

    e ist Evidenz für C, wenn alle x_i mit Evidenz für A_i instantiiert werden

Refinement Logik als formaler Kalkül (II)

- **(Verfeinerungs-)Regel (dec, val)**
  - dec Dekomposition: Abbildung von Sequenzen in Liste von Sequenzen (vom Beweisziel in Liste der Teilziele)
  - val Validierung: Abbildung von Liste von Sequenzen und Termen in Term
  - Regeln werden als Regelschemata dargestellt mit Metavariablen als Platzhaltern für Formeln und Evidenzterme
  - Konkrete Regeln entstehen hieraus durch Instantiierung der Metavariablen
- **Beweise**
  - Jede Sequenz $S = H \vdash C$ ist unvollständiger Beweis mit Wurzel S
  - $(S, r, [\pi_1, \ldots, \pi_n])$ ist Beweis mit Wurzelsequenz S, wenn π_i Beweise für alle Sequenzen S_i sind, die durch Anwendung von r auf S entstehen
  - vollständiger Beweis: Beweis ohne unvollständige Teilbeweise
  - Theorem: vollständiger Beweis, dessen Wurzel eine Initialsequenz ist
- **Extraktterm $ext(\pi)$ eines vollständigen Beweises**
  - $ext(S, (dec, val), [\pi_1, \ldots, \pi_n]) = val([S, (S_1, ext(\pi_1)), .., (S_n, ext(\pi_n))])$, wobei S_i Wurzeln der π_i

Definition gilt auch für $\pi = (S, (dec, val), [])$

Korrektheit und Vollständigkeit

- **Gültigkeit von Sequenzen und Formeln**
  - Eine Sequenz ist gültig, wenn es einen Evidenzterm für sie gibt

  $\mapsto$ Eine Formel C ist gültig, wenn die Initialsequenz $\vdash C$ gültig ist
- **Korrektheit einer Regel $r = (dec, val)$**
  - Sind $S_i = H_i \vdash C_i$ Ergebnis der Anwendung von dec auf $S = H \vdash C$ und c_i Evidenzterme für S_i, dann ist $val(S, (S_i, c_i))$ Evidenzterm für S
- **Refinement Logik ist korrekt**
  - Alle Regeln der Refinement Logik sind korrekt
  - Alle Theoreme der Refinement Logik haben gültige Formeln als Wurzeln

  $\mapsto$ Alle beweisbaren Formeln sind gültig
- **Refinement Logik ist vollständig**
  - Für jede gültige Formel C gibt es ein Theorem mit Wurzel $\vdash C$

  $\mapsto$ Alle gültigen Formeln sind beweisbar

Definitorische Erweiterung

- **Konservative Erweiterung der Objektsprache**
  - Neues Konstrukte sind definitorische Abkürzung für existierende objektsprachliche Ausdrücke (ggf. mit Parametern)
  - Beispiel: Äquivalenz in der Prädikatenlogik

    $$A \Leftrightarrow B \quad \equiv \quad (A \Rightarrow B) \wedge (B \Rightarrow A)$$

  - Bedeutung ergibt sich aus Semantik bestehender Konstrukte
  - Auffalten der Definition in Beweisen ersetzt linke durch rechte Seite

    $$\frac{H \vdash C \quad \texttt{ev} = c}{H \vdash C[rhs/lhs] \quad \texttt{ev} = c} \quad \texttt{unfold } \textit{name}$$

- **Erlaubt kleinen Grundformalismus**
  - Einfache Syntax, Semantik und Inferenzsystem
  - Eigenschaften leicht beweisbar
- **Erhöht Flexibilität des Formalismus**
  - Erlaubt freiere Syntax und umfangreiche formale Sprache

Taktiken und Beweisstrategien

- **Konservative Erweiterung des Regelsystems**
  - Taktiken sind Abkürzungen für komplexe Regelanwendungen
  - Einfache Taktiksprache unterstützt Verknüpfung von Regeln

    r_1 `THEN` r_2: Führe Regel r_2 direkt nach r_1 aus

    r_1 `ORELSE` r_2: Führe Regel r_2 aus, wenn r_1 nicht anwendbar ist

    `Repeat` r: Führe Regel r solange wie möglich aus
  - Ermöglicht kürzere und besser strukturierte Beweise

$\vdash\ P \Rightarrow (Q \Rightarrow (P \wedge Q))$	`BY Repeat impliesR`
`1.` $p{:}P, q{:}Q \vdash P \wedge Q$	`BY andR THEN axiom`

$\vdash\ ((P \vee Q) \wedge ((P \Rightarrow R) \wedge (Q \Rightarrow R))) \Rightarrow R$	`BY impliesR THEN Repeat andL`
`1.` $z{:}P \vee Q, g{:}P \Rightarrow R, h{:}Q \Rightarrow R \vdash R$	`BY orL`
`1.1.` $p{:}P, g{:}P \Rightarrow R, h{:}Q \Rightarrow R \vdash R$	`BY impliesL` g `THEN axiom`
`1.2.` $q{:}Q, g{:}P \Rightarrow R, h{:}Q \Rightarrow R \vdash R$	`BY impliesL` h `THEN axiom`

  - Elementarer Beweis und Extraktterm durch Expansion rekonstruierbar
- **Taktiksprache formalisiert Metasprache der Logik** (ML)
  - Unterstützt Formulierung komplexer Taktiken und Strategien durch Analyse des Beweisziels für gezielte Regelauswahl (ALuP II)

Sequenzenkalkül: Beweismethodik

- **Kalkül garantiert Korrektheit formaler Beweise**
  - Kalkül ist selbst keine Methode um Beweise zu finden
- **Es gibt Leitlinien für erfolgreiche Beweissuche**
  - Versuche vorrangig Zweige abzuschließen (`axiom`)
  - Verwende Dekompositionsregeln, die Formeln äquivalent aufbrechen
  - Verwende `orL` vor `orR1` / `orR2`
  - Verwende `exL` und `allR` vor `exR` und `allL`
  - Wähle anwendbare Regel, welche die wenigsten Teilziele erzeugt

  Methodik ist als Taktik programmierbar (ALuP II)
- **Beweismethodik läßt Fragen offen**
  - Auswahl der Substitution für Quantoren erfordert "Vorausschau"
  - Maschinennahe Methoden finden Substitution durch Unifikation

  Mehr dazu in der Vorlesung "Inferenzmethoden"

Alternative Vorgehensweisen

- **Semantikdefinition durch Interpretation**
  - Interpretation von Variablen, Funktionen, Prädikaten in Zielsprache
  - Homomorphe Fortsetzung der Interpretation auf komplexe Formeln
  - Gültigkeit ist "Wahrheit" unter allen möglichen Interpretationen

    Benötigt Erklärung der Bedeutung der Zielsprache
- **Alternative Beweiskalküle**
  - Axiom-orientierte Frege–Hilbert–Kalküle
    Sehr mächtig, nur eine Regel, aufwendige Beweissuche
  - Natürliches Schließen, Sequenzenkalküle
    Synthetische Vorgehensweise, gut für Beweispräsentation
  - (Analytische) Tableaux-Kalküle
    kompakte, evidenzlose Version der Refinement Logik
  - Maschinennahe Resolutions-/Konnektionskalküle
    Gut geeignet für automatische Beweissuche, schwer lesbare Beweise
- **Ausdrucksstärkere Logiken**
  - Gleichheit, Sorten, Arithmetik, Logik höherer Stufe