Automatisierte Logik und Programmierung

Lektion 15

Beweisautomatisierung für die Logik erster Stufe

1. Taktische Beweismethoden
2. Maschinennahe Beweistechniken
3. JProver: Integration externer Prozeduren

Beweisführung in Logik erster Stufe

- **Interaktive Anwendung logischer Regeln**
  - Benutzer gibt Regeln des Sequenzenkalküls und Parameter an
  - System führt Regeln aus und liefert Teilziele

Beweisführung in Logik erster Stufe

- **Interaktive Anwendung logischer Regeln**
  – Benutzer gibt Regeln des Sequenzenkalküls und Parameter an
  – System führt Regeln aus und liefert Teilziele

  Mühsam, aber sicher

Beweisführung in Logik erster Stufe

- **Interaktive Anwendung logischer Regeln**
  - Benutzer gibt Regeln des Sequenzenkalküls und Parameter an
  - System führt Regeln aus und liefert Teilziele

  Mühsam, aber sicher

- **Taktikbasierte Beweissuche**
  - Taktik sucht nach anwendbaren Regeln
  - Analyse von Konklusion & Hypothesen zur Bestimmung von Parametern

Beweisführung in Logik erster Stufe

- **Interaktive Anwendung logischer Regeln**
  – Benutzer gibt Regeln des Sequenzenkalküls und Parameter an
  – System führt Regeln aus und liefert Teilziele

  Mühsam, aber sicher

- **Taktikbasierte Beweissuche**
  – Taktik sucht nach anwendbaren Regeln
  – Analyse von Konklusion & Hypothesen zur Bestimmung von Parametern

  Hilfreich in der Praxis aber unvollständig wegen begrenzter Vorausschau

Beweisführung in Logik erster Stufe

- **Interaktive Anwendung logischer Regeln**
  – Benutzer gibt Regeln des Sequenzenkalküls und Parameter an
  – System führt Regeln aus und liefert Teilziele

  Mühsam, aber sicher

- **Taktikbasierte Beweissuche**
  – Taktik sucht nach anwendbaren Regeln
  – Analyse von Konklusion & Hypothesen zur Bestimmung von Parametern

  Hilfreich in der Praxis aber unvollständig wegen begrenzter Vorausschau

- **Vollautomatische Beweisverfahren** $\mapsto$ CADE, TABLEAUX, …
  – Transformation einer Sequenz in effiziente Datenstruktur
  – Charakterisierung gültiger Sequenzen durch Eigenschaften dieser Struktur
  – Beweiser benutzt Standardverfahren zur Überprüfung der Eigenschaften

Beweisführung in Logik erster Stufe

- **Interaktive Anwendung logischer Regeln**
  - Benutzer gibt Regeln des Sequenzenkalküls und Parameter an
  - System führt Regeln aus und liefert Teilziele

  Mühsam, aber sicher

- **Taktikbasierte Beweissuche**
  - Taktik sucht nach anwendbaren Regeln
  - Analyse von Konklusion & Hypothesen zur Bestimmung von Parametern

  Hilfreich in der Praxis aber unvollständig wegen begrenzter Vorausschau

- **Vollautomatische Beweisverfahren** $\mapsto$ CADE, TABLEAUX, …
  - Transformation einer Sequenz in effiziente Datenstruktur
  - Charakterisierung gültiger Sequenzen durch Eigenschaften dieser Struktur
  - Beweiser benutzt Standardverfahren zur Überprüfung der Eigenschaften
  - Viele "Standalone" Methoden für klassische Logik
  - Wenige Verfahren können auf konstruktive Logik erweitert werden

Beweisführung in Logik erster Stufe

- **Interaktive Anwendung logischer Regeln**
  - Benutzer gibt Regeln des Sequenzenkalküls und Parameter an
  - System führt Regeln aus und liefert Teilziele

  Mühsam, aber sicher

- **Taktikbasierte Beweissuche**
  - Taktik sucht nach anwendbaren Regeln
  - Analyse von Konklusion & Hypothesen zur Bestimmung von Parametern

  Hilfreich in der Praxis aber unvollständig wegen begrenzter Vorausschau

- **Vollautomatische Beweisverfahren** $\mapsto$ CADE, TABLEAUX, …
  - Transformation einer Sequenz in effiziente Datenstruktur
  - Charakterisierung gültiger Sequenzen durch Eigenschaften dieser Struktur
  - Beweiser benutzt Standardverfahren zur Überprüfung der Eigenschaften
  - Viele “Standalone” Methoden für klassische Logik
  - Wenige Verfahren können auf konstruktive Logik erweitert werden

  Integration aufwendig, da Konsistenzcheck oder Beweisterme erforderlich

Schrittweise Steigerung des Automatisierungsgrades

Schrittweise Steigerung des Automatisierungsgrades

1. **Interaktion mit Regeln der Logik erster Stufe**
   – Konnektive und Quantoren definiert über Curry-Howard Isomorphie

Schrittweise Steigerung des Automatisierungsgrades

1. Interaktion mit Regeln der Logik erster Stufe

– Konnektive und Quantoren definiert über Curry-Howard Isomorphie

– Logikregeln implementiert als Dekomposition + Wohlgeformtheitsprüfung

```
let allR = D 0 THENW Auto
```

Schrittweise Steigerung des Automatisierungsgrades

1. Interaktion mit Regeln der Logik erster Stufe

– Konnektive und Quantoren definiert über Curry-Howard Isomorphie

– Logikregeln implementiert als Dekomposition + Wohlgeformtheitsprüfung

```
let allR = D 0 THENW Auto
```

– Tactical `TryOnC` verhindert Anwendung von Regeln auf unpassende Ziele

```
let andR = TryOnC (D 0) is_and_term
 and orR1 = TryOnC (Sel 1 (D 0) THENW Auto) is_or_term
 and orR2 = TryOnC (Sel 2 (D 0) THENW Auto) is_or_term
 and impR = TryOnC (D 0 THENW Auto) is_imp_term
           ⋮
```

Schrittweise Steigerung des Automatisierungsgrades

1. Interaktion mit Regeln der Logik erster Stufe

– Konnektive und Quantoren definiert über Curry-Howard Isomorphie

– Logikregeln implementiert als Dekomposition + Wohlgeformtheitsprüfung

```
let allR = D 0 THENW Auto
```

– Tactical `TryOnC` verhindert Anwendung von Regeln auf unpassende Ziele

```
let andR = TryOnC (D 0) is_and_term
 and orR1 = TryOnC (Sel 1 (D 0) THENW Auto) is_or_term
 and orR2 = TryOnC (Sel 2 (D 0) THENW Auto) is_or_term
 and impR = TryOnC (D 0 THENW Auto) is_imp_term
          ⋮
```

– Tactical `Run` erzeugt gekapselte Varianten der Regeln

`andI`, `orI1`, `orI2`, `impI`, ..., `andE` i, `orE` i, `impE` i, ...,

Regeln werden bei Inspektion interner Beweise nicht aufgefaltet

2. Bestimmung anwendbarer Regeln

2. Bestimmung anwendbarer Regeln

– Tactical `TryAllHyps` wendet Taktik auf erste passende Hypothese an

2. Bestimmung anwendbarer Regeln

– Tactical `TryAllHyps` wendet Taktik auf erste passende Hypothese an

```
let contradiction = TryAllHyps falseE is_false_term
let conjunctionE = TryAllHyps andE is_and_term
let disjunctionE = TryAllHyps orE is_or_term
let existentialE = TryAllHyps exE is_ex_term
```

2. Bestimmung anwendbarer Regeln

– Tactical `TryAllHyps` wendet Taktik auf erste passende Hypothese an

```
let contradiction = TryAllHyps falseE is_false_term
let conjunctionE = TryAllHyps andE is_and_term
let disjunctionE = TryAllHyps orE is_or_term
let existentialE = TryAllHyps exE is_ex_term
```

– Einführungsregeln können namentlich bestimmt werden

```
let nondangerousI pf =
  let kind = operator_id_of_term (conclusion pf)
  in
   if mem kind [`all`;`not`;`implies`;`rev_implies`;`iff`;`and`]
    then Run (kind ^ `R`) pf
    else failwith `tactic inappropriate`
  ;;
```

Taktikbasierte Beweisführung III

3. Verkettung von Implikationen & Äquivalenzen

– Tactical `Chain` wendet Taktik t auf ausgewählte Hypothesen an und schließt mit Basistaktik ab

Taktikbasierte Beweisführung III

3. Verkettung von Implikationen & Äquivalenzen

– Tactical `Chain` wendet Taktik t auf ausgewählte Hypothesen an und schließt mit Basistaktik ab

```
let Chain t hyps groundtac =
  letrec chainfor i =
    groundtac
    ORELSE
    if i = 0 then Id
      else TryOn hyps (\hyp. t hyp THEN (Complete (chainfor (i-1))))
  in
    chain_for chain_limit
 ;;
```

3. Verkettung von Implikationen & Äquivalenzen

– Tactical `Chain` wendet Taktik t auf ausgewählte Hypothesen an und schließt mit Basistaktik ab

```
let Chain t hyps groundtac =
   letrec chainfor i =
     groundtac
     ORELSE
     if i = 0 then Id
       else TryOn hyps (\hyp. t hyp THEN (Complete (chainfor (i-1))))
   in
     chain_for chain_limit
 ;;
```

```
let imp_chain pf =
    Chain impE (select_hyps is_imp_term pf) Hypothesis pf ;;
let not_chain =
    TryAllHyps (\pos. notE pos THEN imp_chain) is_not_term ;;
let iff_chain =
    TryAllHyps (\pos. (iffE pos THEN (imp_chain ORELSE not_chain))
                ORELSE (iffE_b pos THEN (imp_chain ORELSE not_chain))
                ) is_iff_term;;
```

Taktikbasierte Beweisführung IV

Metalevel Analyse zur Instantiierung von Quantoren

```
let match_subEx quantified_term assumption  =
   letrec match_sub_aux vars exprop =
     map (\var.assoc var (match vars exprop assumption)) (rev vars)
    ? let var,T,prop = dest_exists exprop in match_sub_aux (var.vars) prop
   in
      match_sub_aux [] quantified_term;;

letrec exIon terms pf =
    let t.rest = terms in (exI t THEN exIon rest) pf  ?  Id pf;;

let InstantiateEx =
  let InstEx_aux pos pf =
     let sigma = match_subEx (conclusion pf) (type_of_hyp pos pf)   in
        (exIon (map snd sigma) THEN (NthHyp pos)) pf
  in
     OnSomeHyp InstEx_aux;;

let InstantiateAll =
  let InstAll_aux pos pf =
     let sigma = match_subAll (type_of_hyp pos pf) (conclusion pf) in
          (allEon pos (map snd sigma) THEN (OnLastHyp hypothesis)) pf
  in
      TryAllHyps InstAll_aux  is_all_term;;
```

Ein einfacher taktikbasierter Beweiser

Sortiere Regelanwendungen nach Aufwand für Beweissuche

Ein einfacher taktikbasierter Beweiser

Sortiere Regelanwendungen nach Aufwand für Beweissuche

```
let simple_prover = Repeat
                    (      Hypothesis
                    ORELSE contradiction
```

Ein einfacher taktikbasierter Beweiser

Sortiere Regelanwendungen nach Aufwand für Beweissuche

```
let simple_prover = Repeat
                    (       Hypothesis
                    ORELSE contradiction
                    ORELSE InstantiateAll
                    ORELSE InstantiateEx
```

Ein einfacher taktikbasierter Beweiser

Sortiere Regelanwendungen nach Aufwand für Beweissuche

```
let simple_prover = Repeat
                    (       Hypothesis
                     ORELSE contradiction
                     ORELSE InstantiateAll
                     ORELSE InstantiateEx
                     ORELSE conjunctionE
                     ORELSE existentialE
                     ORELSE nondangerousI
```

Ein einfacher taktikbasierter Beweiser

Sortiere Regelanwendungen nach Aufwand für Beweissuche

```
let simple_prover = Repeat
                    (      Hypothesis
                    ORELSE contradiction
                    ORELSE InstantiateAll
                    ORELSE InstantiateEx
                    ORELSE conjunctionE
                    ORELSE existentialE
                    ORELSE nondangerousI
                    ORELSE disjunctionE
```

Ein einfacher taktikbasierter Beweiser

Sortiere Regelanwendungen nach Aufwand für Beweissuche

```
let simple_prover = Repeat
                    (       Hypothesis
                    ORELSE contradiction
                    ORELSE InstantiateAll
                    ORELSE InstantiateEx
                    ORELSE conjunctionE
                    ORELSE existentialE
                    ORELSE nondangerousI
                    ORELSE disjunctionE
                    ORELSE not_chain
                    ORELSE iff_chain
                    ORELSE imp_chain
                    );;
```

Ein einfacher taktikbasierter Beweiser

Sortiere Regelanwendungen nach Aufwand für Beweissuche

```
let simple_prover = Repeat
                    (       Hypothesis
                    ORELSE contradiction
                    ORELSE InstantiateAll
                    ORELSE InstantiateEx
                    ORELSE conjunctionE
                    ORELSE existentialE
                    ORELSE nondangerousI
                    ORELSE disjunctionE
                    ORELSE not_chain
                    ORELSE iff_chain
                    ORELSE imp_chain
                    );;

letrec prover = simple_prover
```

Ein einfacher taktikbasierter Beweiser

Sortiere Regelanwendungen nach Aufwand für Beweissuche

```
let simple_prover = Repeat
                    (      Hypothesis
                    ORELSE contradiction
                    ORELSE InstantiateAll
                    ORELSE InstantiateEx
                    ORELSE conjunctionE
                    ORELSE existentialE
                    ORELSE nondangerousI
                    ORELSE disjunctionE
                    ORELSE not_chain
                    ORELSE iff_chain
                    ORELSE imp_chain
                    );;

letrec prover = simple_prover
                THEN Try (        Complete (orI1 THEN prover)
                          ORELSE (Complete (orI2 THEN prover)))
                ;;
```

Mögliche Verbesserungen des Beweisers

- **Verbessertes Matching**
  - Matching mit Teiltermen von Konjunktionen in Hypothesen
  - Matching mit Teiltermen von Disjunktionen in der Konklusion
  - Gleichzeitige Analyse von Quantoren in Hypothesen und Konklusion
  - Behandlung verschachtelt wechselnder Quantoren

$\vdots$

Mögliche Verbesserungen des Beweisers

- **Verbessertes Matching**
  - Matching mit Teiltermen von Konjunktionen in Hypothesen
  - Matching mit Teiltermen von Disjunktionen in der Konklusion
  - Gleichzeitige Analyse von Quantoren in Hypothesen und Konklusion
  - Behandlung verschachtelt wechselnder Quantoren

    ⋮

- **Zielgerichtete Verkettung**
  - Auswahl relevanter Implikationen & Äquivalenzen durch Matching
  - Analyse von Teilen der Prämissen von Implikationen

    ⋮

Mögliche Verbesserungen des Beweisers

- **Verbessertes Matching**
  - Matching mit Teiltermen von Konjunktionen in Hypothesen
  - Matching mit Teiltermen von Disjunktionen in der Konklusion
  - Gleichzeitige Analyse von Quantoren in Hypothesen und Konklusion
  - Behandlung verschachtelt wechselnder Quantoren

    ⋮

- **Zielgerichtete Verkettung**
  - Auswahl relevanter Implikationen & Äquivalenzen durch Matching
  - Analyse von Teilen der Prämissen von Implikationen

    ⋮

- **Taktikbasiertes Beweisen hat Grenzen**
  - Regeln `exR`, `allL`, … benötigen Parameter
  - Regeln `orR1`, `orR2`, … benötigen Steuerung
  - Auswahl der richtigen Regel benötigt “Planung” des Beweises
  - Vollständige Beweissuche braucht “Vorausschau” durch Metalevel Analyse

Maschinennahe Beweistechniken

- **Ziel: einfache und schnelle Suchtechnik**
  - Verzicht auf intuitives Verständnis im Beweissuchverfahren
  - Maschinennahe Charakterisierung logischer Gültigkeit
  - Effiziente, "low-level" Suchstrategien auf Basis spezieller Datenstrukturen

Maschinennahe Beweistechniken

- **Ziel: einfache und schnelle Suchtechnik**
  - Verzicht auf intuitives Verständnis im Beweissuchverfahren
  - Maschinennahe Charakterisierung logischer Gültigkeit
  - Effiziente, “low-level” Suchstrategien auf Basis spezieller Datenstrukturen

- **Viele unabhängig entstandene Verfahren**
  - **Resolution**: Widerlegungsverfahren für Formeln in DNF $\mapsto$ Prolog
    - Verschmelze Klauseln mit “komplementären” Literalen
    - Komplementaritätstest erster Stufe benötigt **Unifikation**
    - Ziel ist Herleitung der leeren Klausel

Maschinennahe Beweistechniken

- **Ziel: einfache und schnelle Suchtechnik**
  – Verzicht auf intuitives Verständnis im Beweissuchverfahren
  – Maschinennahe Charakterisierung logischer Gültigkeit
  – Effiziente, “low-level” Suchstrategien auf Basis spezieller Datenstrukturen

- **Viele unabhängig entstandene Verfahren**
  – **Resolution**: Widerlegungsverfahren für Formeln in DNF $\mapsto$ Prolog
    · Verschmelze Klauseln mit “komplementären” Literalen
    · Komplementaritätstest erster Stufe benötigt **Unifikation**
    · Ziel ist Herleitung der leeren Klausel
  – **Matrixmethoden**: Kompakte Repräsentation von Suchbäumen
    · Matrix repräsentiert Verzweigungsstruktur von Beweisbäumen
    · Teste, ob alle Pfade komplementäre Literale enthalten

Maschinennahe Beweistechniken

- **Ziel: einfache und schnelle Suchtechnik**
  - Verzicht auf intuitives Verständnis im Beweissuchverfahren
  - Maschinennahe Charakterisierung logischer Gültigkeit
  - Effiziente, “low-level” Suchstrategien auf Basis spezieller Datenstrukturen

- **Viele unabhängig entstandene Verfahren**
  - **Resolution**: Widerlegungsverfahren für Formeln in DNF $\mapsto$ Prolog
    - Verschmelze Klauseln mit “komplementären” Literalen
    - Komplementaritätstest erster Stufe benötigt **Unifikation**
    - Ziel ist Herleitung der leeren Klausel
  - **Matrixmethoden**: Kompakte Repräsentation von Suchbäumen
    - Matrix repräsentiert Verzweigungsstruktur von Beweisbäumen
    - Teste, ob alle Pfade komplementäre Literale enthalten
  - **Inverse Methode**: Ähnlich zur Resolution
  - **Modellelimination**: Zeige, daß kein Modell die Formel falsifizieren kann

Maschinennahe Beweistechniken

- **Ziel: einfache und schnelle Suchtechnik**
  – Verzicht auf intuitives Verständnis im Beweissuchverfahren
  – Maschinennahe Charakterisierung logischer Gültigkeit
  – Effiziente, “low-level” Suchstrategien auf Basis spezieller Datenstrukturen

- **Viele unabhängig entstandene Verfahren**
  – **Resolution**: Widerlegungsverfahren für Formeln in DNF $\mapsto$ Prolog
    · Verschmelze Klauseln mit “komplementären” Literalen
    · Komplementaritätstest erster Stufe benötigt **Unifikation**
    · Ziel ist Herleitung der leeren Klausel
  – **Matrixmethoden**: Kompakte Repräsentation von Suchbäumen
    · Matrix repräsentiert Verzweigungsstruktur von Beweisbäumen
    · Teste, ob alle Pfade komplementäre Literale enthalten
  – **Inverse Methode**: Ähnlich zur Resolution
  – **Modellelimination**: Zeige, daß kein Modell die Formel falsifizieren kann
  – **Davis Putnam**: Iterative Anwendung von Aufspaltung und Reduktion
    · Schnellstes Verfahren für Aussagenlogik, nicht erweiterbar

Maschinennahe Beweistechniken

- **Ziel: einfache und schnelle Suchtechnik**
  - Verzicht auf intuitives Verständnis im Beweissuchverfahren
  - Maschinennahe Charakterisierung logischer Gültigkeit
  - Effiziente, "low-level" Suchstrategien auf Basis spezieller Datenstrukturen

- **Viele unabhängig entstandene Verfahren**
  - **Resolution**: Widerlegungsverfahren für Formeln in DNF $\mapsto$ Prolog
    - Verschmelze Klauseln mit "komplementären" Literalen
    - Komplementaritätstest erster Stufe benötigt **Unifikation**
    - Ziel ist Herleitung der leeren Klausel
  - **Matrixmethoden**: Kompakte Repräsentation von Suchbäumen
    - Matrix repräsentiert Verzweigungsstruktur von Beweisbäumen
    - Teste, ob alle Pfade komplementäre Literale enthalten
  - **Inverse Methode**: Ähnlich zur Resolution
  - **Modellelimination**: Zeige, daß kein Modell die Formel falsifizieren kann
  - **Davis Putnam**: Iterative Anwendung von Aufspaltung und Reduktion
    - Schnellstes Verfahren für Aussagenlogik, nicht erweiterbar
  - **Wenige Verfahren liefern Beweisterme**

Tableauxmethodik

- **Sequenzenbeweise enthalten viel Redundanz**
  – Jeder Knoten enthält alle gültigen Annahmen und die Konklusion
  – Inferenzregeln basieren auf logischen Konnektiven und Quantoren

Tableauxmethodik

- **Sequenzenbeweise enthalten viel Redundanz**
  - Jeder Knoten enthält alle gültigen Annahmen und die Konklusion
  - Inferenzregeln basieren auf logischen Konnektiven und Quantoren
- **Tableauxmethode verkürzt Sequenzenbeweise**

Tableauxmethodik

- **Sequenzenbeweise enthalten viel Redundanz**
  – Jeder Knoten enthält alle gültigen Annahmen und die Konklusion
  – Inferenzregeln basieren auf logischen Konnektiven und Quantoren
- **Tableauxmethode verkürzt Sequenzenbeweise**
  – **Polarität** kennzeichnet Unterschied zwischen Annahmen und Konklusion

Tableauxmethodik

- **Sequenzenbeweise enthalten viel Redundanz**
  - Jeder Knoten enthält alle gültigen Annahmen und die Konklusion
  - Inferenzregeln basieren auf logischen Konnektiven und Quantoren
- **Tableauxmethode verkürzt Sequenzenbeweise**
  - **Polarität** kennzeichnet Unterschied zwischen Annahmen und Konklusion
  - Inferenzregeln werden zu Klassen ähnlicher Struktur zusammengefaßt

`andL` i	$\Gamma, A \wedge B, \Delta \vdash C$	$\Gamma \vdash A \wedge B$	`andR`
	$\Gamma, A, B, \Delta \vdash C$	$\Gamma \vdash A$	
		$\Gamma \vdash B$	
`orL` i	$\Gamma, A \vee B, \Delta \vdash C$	$\Gamma \vdash A \vee B$	`orR1`
	$\Gamma, A, \Delta \vdash C$	$\Gamma \vdash A$	
	$\Gamma, B, \Delta \vdash C$	$\Gamma \vdash A \vee B$	`orR2`
		$\Gamma \vdash B$	

  - `andL` und `orR`: Dekomposition liefert ein Teilziel **Typ α**

Tableauxmethodik

- **Sequenzenbeweise enthalten viel Redundanz**
  - Jeder Knoten enthält alle gültigen Annahmen und die Konklusion
  - Inferenzregeln basieren auf logischen Konnektiven und Quantoren
- **Tableauxmethode verkürzt Sequenzenbeweise**
  - **Polarität** kennzeichnet Unterschied zwischen Annahmen und Konklusion
  - Inferenzregeln werden zu Klassen ähnlicher Struktur zusammengefaßt

`andL` i	$\Gamma,\, A \wedge B,\, \Delta \vdash C$	$\Gamma \vdash A \wedge B$	`andR`
	$\Gamma,\, A,\, B,\, \Delta \vdash C$	$\Gamma \vdash A$	
		$\Gamma \vdash B$	
`orL` i	$\Gamma,\, A \vee B,\, \Delta \vdash C$	$\Gamma \vdash A \vee B$	`orR1`
	$\Gamma,\, A,\, \Delta \vdash C$	$\Gamma \vdash A$	
	$\Gamma,\, B,\, \Delta \vdash C$	$\Gamma \vdash A \vee B$	`orR2`
		$\Gamma \vdash B$	

  - `andL` und `orR`: Dekomposition liefert ein Teilziel **Typ α**
  - `andR` und `orL`: Dekomposition verzeigt Beweis **Typ β**

TABLEAUXMETHODIK

- **Sequenzenbeweise enthalten viel Redundanz**
  - Jeder Knoten enthält alle gültigen Annahmen und die Konklusion
  - Inferenzregeln basieren auf logischen Konnektiven und Quantoren
- **Tableauxmethode verkürzt Sequenzenbeweise**
  - **Polarität** kennzeichnet Unterschied zwischen Annahmen und Konklusion
  - Inferenzregeln werden zu Klassen ähnlicher Struktur zusammengefaßt

`andL` i	$\Gamma, A \wedge B, \Delta \vdash C$	$\Gamma \vdash A \wedge B$	`andR`
	$\Gamma, A, B, \Delta \vdash C$	$\Gamma \vdash A$	
		$\Gamma \vdash B$	
`orL` i	$\Gamma, A \vee B, \Delta \vdash C$	$\Gamma \vdash A \vee B$	`orR1`
	$\Gamma, A, \Delta \vdash C$	$\Gamma \vdash A$	
	$\Gamma, B, \Delta \vdash C$	$\Gamma \vdash A \vee B$	`orR2`
		$\Gamma \vdash B$	

  - **`andL`** und **`orR`**: Dekomposition liefert ein Teilziel — **Typ α**
  - **`andR`** und **`orL`**: Dekomposition verzeigt Beweis — **Typ β**
  - **`allL`** und **`exR`**: Dekomposition instantiiert Variable mit Term — **Typ γ**

Tableauxmethodik

- **Sequenzenbeweise enthalten viel Redundanz**
  – Jeder Knoten enthält alle gültigen Annahmen und die Konklusion
  – Inferenzregeln basieren auf logischen Konnektiven und Quantoren

- **Tableauxmethode verkürzt Sequenzenbeweise**
  – **Polarität** kennzeichnet Unterschied zwischen Annahmen und Konklusion
  – Inferenzregeln werden zu Klassen ähnlicher Struktur zusammengefaßt

`andL` i	$\Gamma,\ A \wedge B,\ \Delta \vdash C$	$\Gamma \vdash A \wedge B$	`andR`
	$\Gamma,\ A,\ B,\ \Delta \vdash C$	$\Gamma \vdash A$	
		$\Gamma \vdash B$	
`orL` i	$\Gamma,\ A \vee B,\ \Delta \vdash C$	$\Gamma \vdash A \vee B$	`orR1`
	$\Gamma,\ A,\ \Delta \vdash C$	$\Gamma \vdash A$	
	$\Gamma,\ B,\ \Delta \vdash C$	$\Gamma \vdash A \vee B$	`orR2`
		$\Gamma \vdash B$	

  – `andL` und `orR`: Dekomposition liefert ein Teilziel **Typ α**
  – `andR` und `orL`: Dekomposition verzeigt Beweis **Typ β**
  – `allL` und `exR`: Dekomposition instantiiert Variable mit Term **Typ γ**
  – `allR` und `exL`: Dekomposition deklariert neue Variable **Typ δ**

Tableauxmethodik

- **Sequenzenbeweise enthalten viel Redundanz**
  - Jeder Knoten enthält alle gültigen Annahmen und die Konklusion
  - Inferenzregeln basieren auf logischen Konnektiven und Quantoren
- **Tableauxmethode verkürzt Sequenzenbeweise**
  - **Polarität** kennzeichnet Unterschied zwischen Annahmen und Konklusion
  - Inferenzregeln werden zu Klassen ähnlicher Struktur zusammengefaßt

`andL` i	$\Gamma, A \wedge B, \Delta \vdash C$ $\quad\Gamma, A, B, \Delta \vdash C$	$\Gamma \vdash A \wedge B$ $\quad\Gamma \vdash A$ $\quad\Gamma \vdash B$	`andR`
`orL` i	$\Gamma, A \vee B, \Delta \vdash C$ $\quad\Gamma, A, \Delta \vdash C$ $\quad\Gamma, B, \Delta \vdash C$	$\Gamma \vdash A \vee B$ $\quad\Gamma \vdash A$	`orR1`
		$\Gamma \vdash A \vee B$ $\quad\Gamma \vdash B$	`orR2`

  - `andL` und `orR`: Dekomposition liefert ein Teilziel **Typ α**
  - `andR` und `orL`: Dekomposition verzeigt Beweis **Typ β**
  - `allL` und `exR`: Dekomposition instantiiert Variable mit Term **Typ γ**
  - `allR` und `exL`: Dekomposition deklariert neue Variable **Typ δ**
  - `hypothesis`: Gleiche Formeln mit anderer Polarität **Komplementarität**

Matrixmethoden: Kompakte Sequenzenbeweise

- **Konstruktion von Beweisbäumen ist aufwendig**
  - Sequenzenbeweise zerlegen Formeln bis `hypothesis` Regel anwendbar
  - Regeln ergänzen Teilformeln in Nachfolgerknoten des Beweises

Matrixmethoden: Kompakte Sequenzenbeweise

- **Konstruktion von Beweisbäumen ist aufwendig**
  - Sequenzenbeweise zerlegen Formeln bis `hypothesis` Regel anwendbar
  - Regeln ergänzen Teilformeln in Nachfolgerknoten des Beweises
- **Kompakte Repräsentation von Beweisbäumen**
  - Formelbaum enthält bereits alle Teilformeln

Matrixmethoden: Kompakte Sequenzenbeweise

- **Konstruktion von Beweisbäumen ist aufwendig**
  - Sequenzenbeweise zerlegen Formeln bis `hypothesis` Regel anwendbar
  - Regeln ergänzen Teilformeln in Nachfolgerknoten des Beweises
- **Kompakte Repräsentation von Beweisbäumen**
  - Formelbaum enthält bereits alle Teilformeln
  - Tableauxtypen und Polaritäten können top-down ergänzt werden
    - Beide hängen nur vom Konnektiv und bisheriger Polarität ab

Matrixmethoden: Kompakte Sequenzenbeweise

- **Konstruktion von Beweisbäumen ist aufwendig**
  - Sequenzenbeweise zerlegen Formeln bis `hypothesis` Regel anwendbar
  - Regeln ergänzen Teilformeln in Nachfolgerknoten des Beweises
- **Kompakte Repräsentation von Beweisbäumen**
  - Formelbaum enthält bereits alle Teilformeln
  - Tableauxtypen und Polaritäten können top-down ergänzt werden
    - Beide hängen nur vom Konnektiv und bisheriger Polarität ab
  - Äste eines Sequenzenbeweises sind durch β-Knoten definiert
  - Teilformeln mit α-Knoten als gemeinsamen Vorgänger erscheinen im gleichen Ast eines Sequenzenbeweises

Matrixmethoden: Kompakte Sequenzenbeweise

- **Konstruktion von Beweisbäumen ist aufwendig**
  – Sequenzenbeweise zerlegen Formeln bis `hypothesis` Regel anwendbar
  – Regeln ergänzen Teilformeln in Nachfolgerknoten des Beweises
- **Kompakte Repräsentation von Beweisbäumen**
  – Formelbaum enthält bereits alle Teilformeln
  – Tableauxtypen und Polaritäten können top-down ergänzt werden
    · Beide hängen nur vom Konnektiv und bisheriger Polarität ab
  – Äste eines Sequenzenbeweises sind durch β-Knoten definiert
  – Teilformeln mit α-Knoten als gemeinsamen Vorgänger erscheinen im gleichen Ast eines Sequenzenbeweises
  – `hypothesis` Regel $\hat{=}$ komplementäre atomare Formeln in "α-Beziehung"

Matrixmethoden: Kompakte Sequenzenbeweise

- **Konstruktion von Beweisbäumen ist aufwendig**
  - Sequenzenbeweise zerlegen Formeln bis `hypothesis` Regel anwendbar
  - Regeln ergänzen Teilformeln in Nachfolgerknoten des Beweises
- **Kompakte Repräsentation von Beweisbäumen**
  - Formelbaum enthält bereits alle Teilformeln
  - Tableauxtypen und Polaritäten können top-down ergänzt werden
    - Beide hängen nur vom Konnektiv und bisheriger Polarität ab
  - Äste eines Sequenzenbeweises sind durch β-Knoten definiert
  - Teilformeln mit α-Knoten als gemeinsamen Vorgänger erscheinen im gleichen Ast eines Sequenzenbeweises
  - `hypothesis` Regel $\hat{=}$ komplementäre atomare Formeln in "α-Beziehung"
- **Einfache Beweismethode**
  - Ordne **Literale** (atomare Formeln) in zweidimensionaler Matrix an
    - Nebeneinander $\hat{=}$ α-Beziehung, übereinander $\hat{=}$ β-Beziehung

Matrixmethoden: Kompakte Sequenzenbeweise

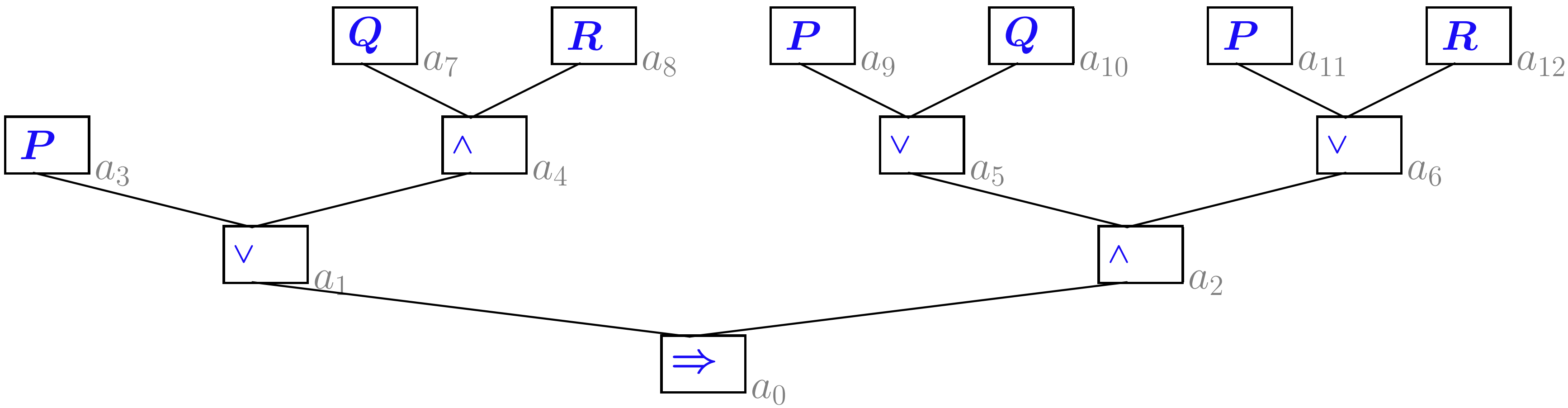
- **Konstruktion von Beweisbäumen ist aufwendig**
  - Sequenzenbeweise zerlegen Formeln bis `hypothesis` Regel anwendbar
  - Regeln ergänzen Teilformeln in Nachfolgerknoten des Beweises
- **Kompakte Repräsentation von Beweisbäumen**
  - Formelbaum enthält bereits alle Teilformeln
  - Tableauxtypen und Polaritäten können top-down ergänzt werden
    - Beide hängen nur vom Konnektiv und bisheriger Polarität ab
  - Äste eines Sequenzenbeweises sind durch β-Knoten definiert
  - Teilformeln mit α-Knoten als gemeinsamen Vorgänger erscheinen im gleichen Ast eines Sequenzenbeweises
  - `hypothesis` Regel $\hat{=}$ komplementäre atomare Formeln in "α-Beziehung"
- **Einfache Beweismethode**
  - Ordne **Literale** (atomare Formeln) in zweidimensionaler Matrix an
    - Nebeneinander $\hat{=}$ α-Beziehung, übereinander $\hat{=}$ β-Beziehung
  - Teste alle Pfade auf Existenz komplementärer Literale

Matrixmethoden: annotierte Formelbäume

$$(P \vee (Q \wedge R)) \Rightarrow ((P \vee Q) \wedge (P \vee R))$$

Matrixmethoden: annotierte Formelbäume

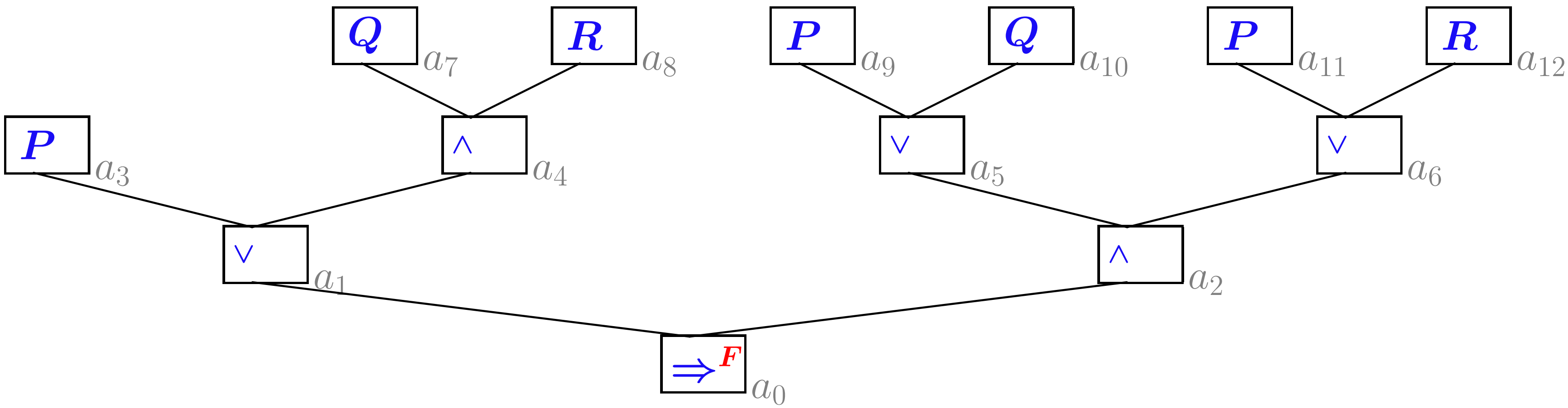
$$(P \vee (Q \wedge R)) \Rightarrow ((P \vee Q) \wedge (P \vee R))$$



Parsen der Formel erzeugt Formelbaum

Matrixmethoden: annotierte Formelbäume

$$(P \vee (Q \wedge R)) \Rightarrow ((P \vee Q) \wedge (P \vee R))$$

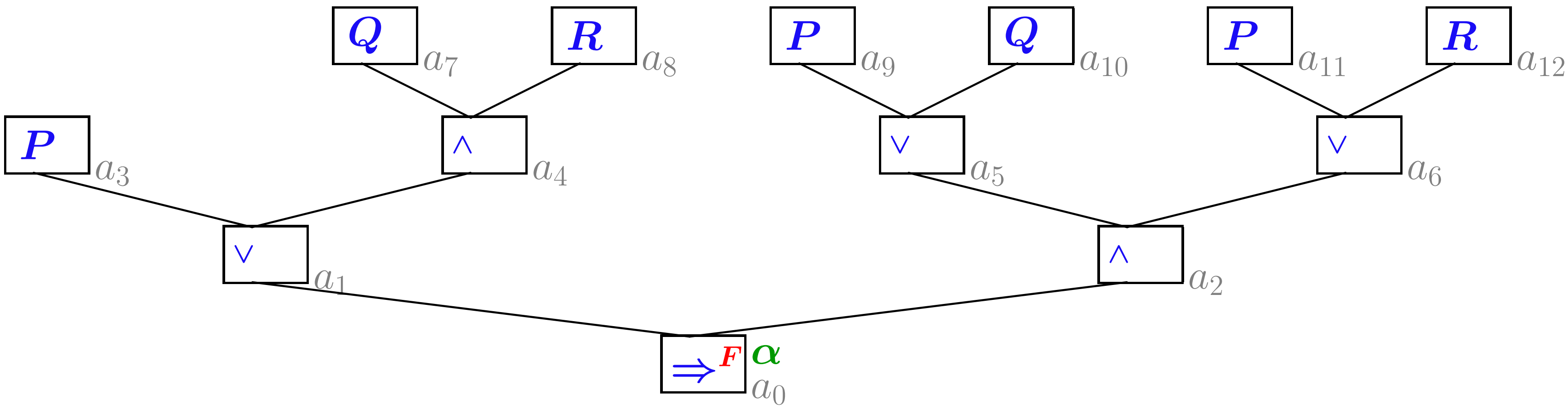


Parsen der Formel erzeugt Formelbaum

- **Zuweisung von Polaritäten:** T ≙ Hypothese, F ≙ Konklusion

Matrixmethoden: annotierte Formelbäume

$$(P \vee (Q \wedge R)) \Rightarrow ((P \vee Q) \wedge (P \vee R))$$

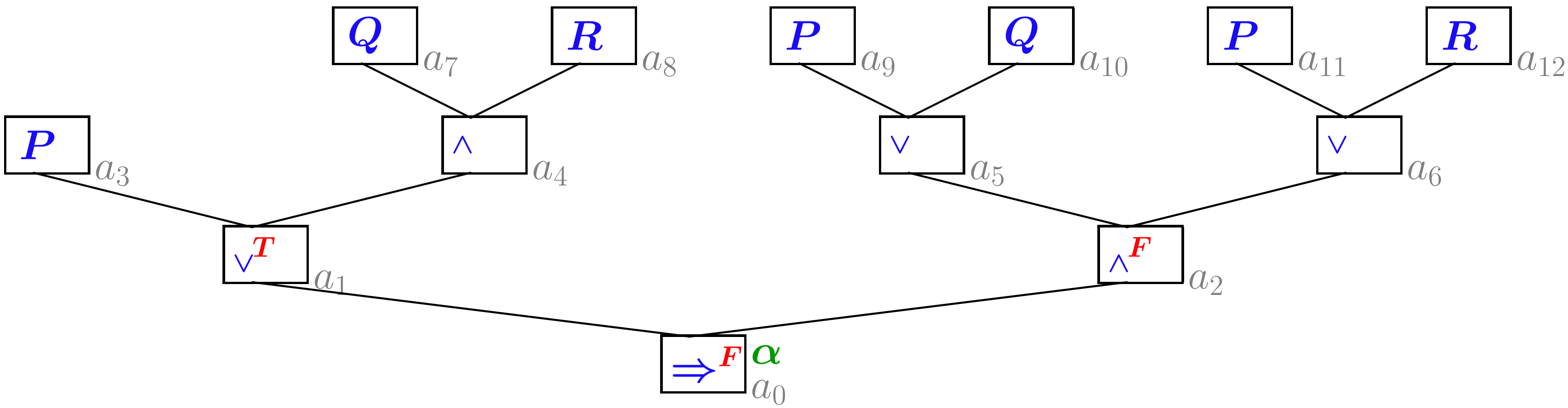


Parsen der Formel erzeugt Formelbaum

- **Zuweisung von Polaritäten:** T ≙ Hypothese, F ≙ Konklusion
- **Bestimmung des Typs:** α ≙ linear, β ≙ Verzweigung

Matrixmethoden: annotierte Formelbäume

$$(P \vee (Q \wedge R)) \Rightarrow ((P \vee Q) \wedge (P \vee R))$$

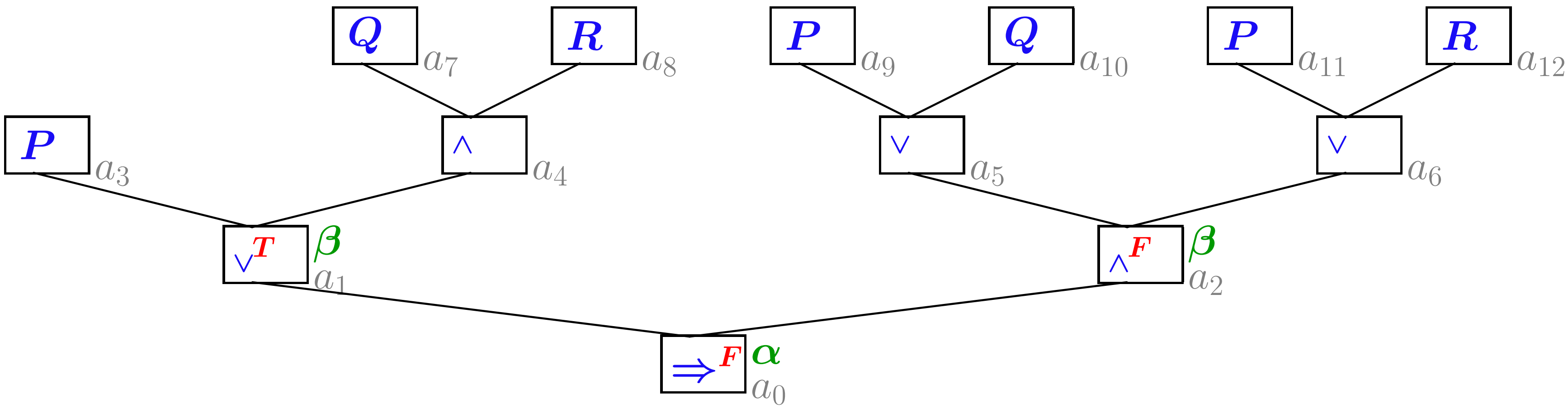


Parsen der Formel erzeugt Formelbaum

- **Zuweisung von Polaritäten:** $\boldsymbol{T}$ $\hat{=}$ Hypothese, $\boldsymbol{F}$ $\hat{=}$ Konklusion
- **Bestimmung des Typs:** $\boldsymbol{\alpha}$ $\hat{=}$ linear, $\boldsymbol{\beta}$ $\hat{=}$ Verzweigung
- **Zuweisung von Polaritäten an Unterformeln**

Matrixmethoden: annotierte Formelbäume

$$(P \vee (Q \wedge R)) \Rightarrow ((P \vee Q) \wedge (P \vee R))$$

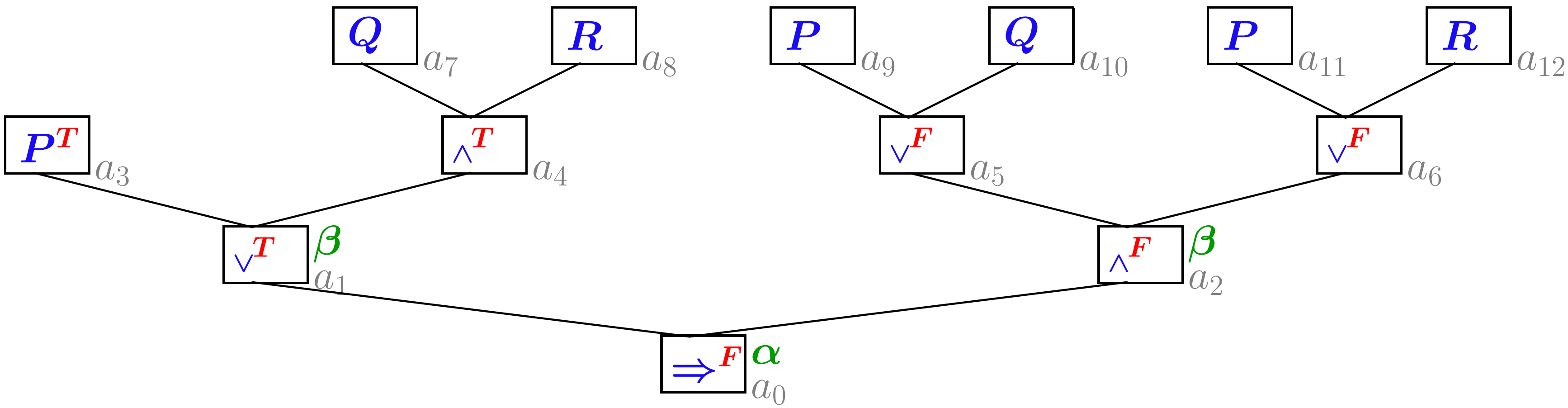


Parsen der Formel erzeugt Formelbaum

- **Zuweisung von Polaritäten:** T $\widehat{=}$ Hypothese, F $\widehat{=}$ Konklusion
- **Bestimmung des Typs:** α $\widehat{=}$ linear, β $\widehat{=}$ Verzweigung
- **Zuweisung von Polaritäten an Unterformeln**
- **Bestimmung des Typs der Unterformeln**

Matrixmethoden: annotierte Formelbäume

$$(P \vee (Q \wedge R)) \Rightarrow ((P \vee Q) \wedge (P \vee R))$$

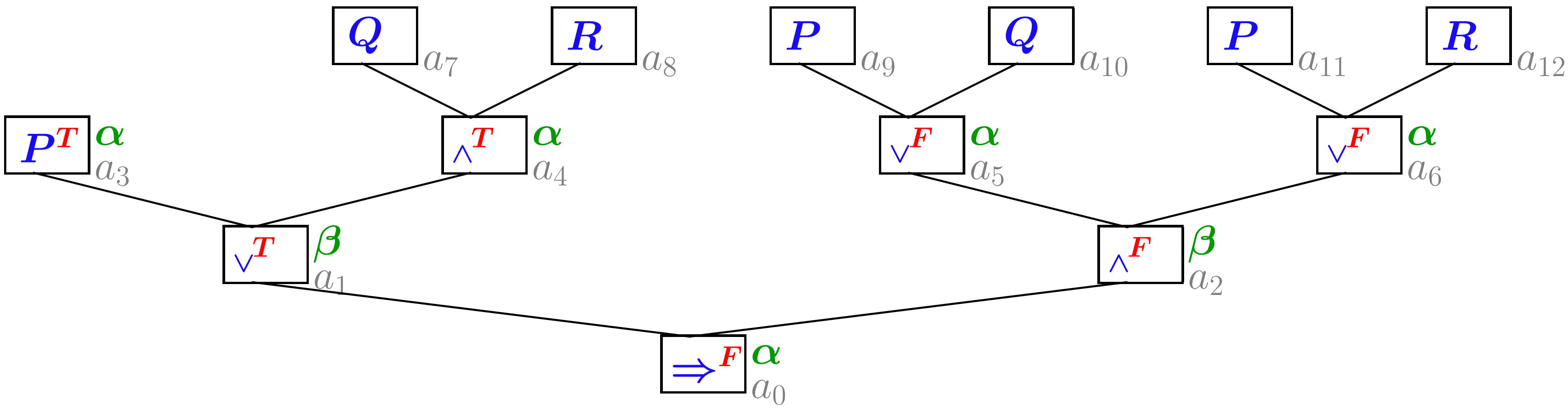


Parsen der Formel erzeugt Formelbaum

- **Zuweisung von Polaritäten:** $T \widehat{=}$ Hypothese, $F \widehat{=}$ Konklusion
- **Bestimmung des Typs:** $\alpha \widehat{=}$ linear, $\beta \widehat{=}$ Verzweigung
- **Zuweisung von Polaritäten an Unterformeln**
- **Bestimmung des Typs der Unterformeln**

Matrixmethoden: annotierte Formelbäume

$$(P \vee (Q \wedge R)) \Rightarrow ((P \vee Q) \wedge (P \vee R))$$

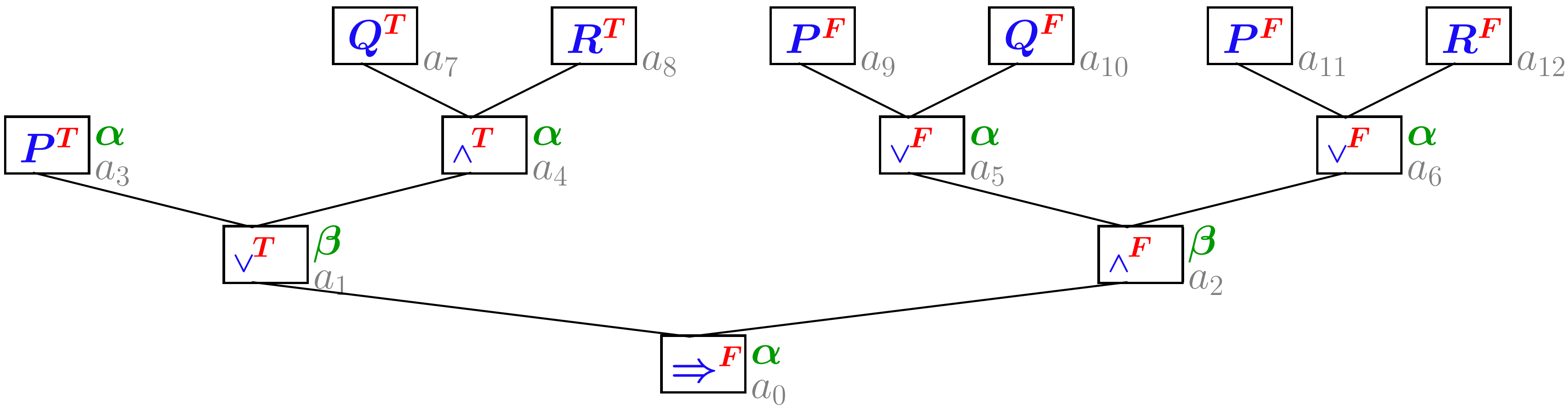


Parsen der Formel erzeugt Formelbaum

- **Zuweisung von Polaritäten:** T ≙ Hypothese, F ≙ Konklusion
- **Bestimmung des Typs:** α ≙ linear, β ≙ Verzweigung
- **Zuweisung von Polaritäten an Unterformeln**
- **Bestimmung des Typs der Unterformeln**

Matrixmethoden: annotierte Formelbäume

$$(P \vee (Q \wedge R)) \Rightarrow ((P \vee Q) \wedge (P \vee R))$$

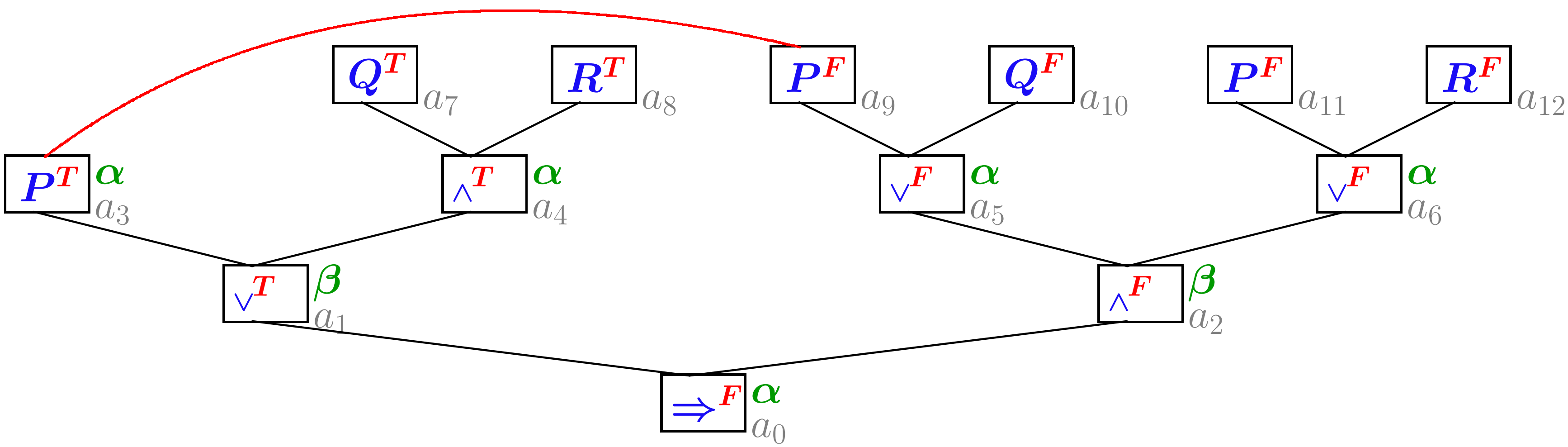


Parsen der Formel erzeugt Formelbaum

- **Zuweisung von Polaritäten:** $\boldsymbol{T}$ $\widehat{=}$ Hypothese, $\boldsymbol{F}$ $\widehat{=}$ Konklusion
- **Bestimmung des Typs:** $\boldsymbol{\alpha}$ $\widehat{=}$ linear, $\boldsymbol{\beta}$ $\widehat{=}$ Verzweigung
- **Zuweisung von Polaritäten an Unterformeln**
- **Bestimmung des Typs der Unterformeln**

Matrixmethoden: annotierte Formelbäume

$$(P \vee (Q \wedge R)) \Rightarrow ((P \vee Q) \wedge (P \vee R))$$

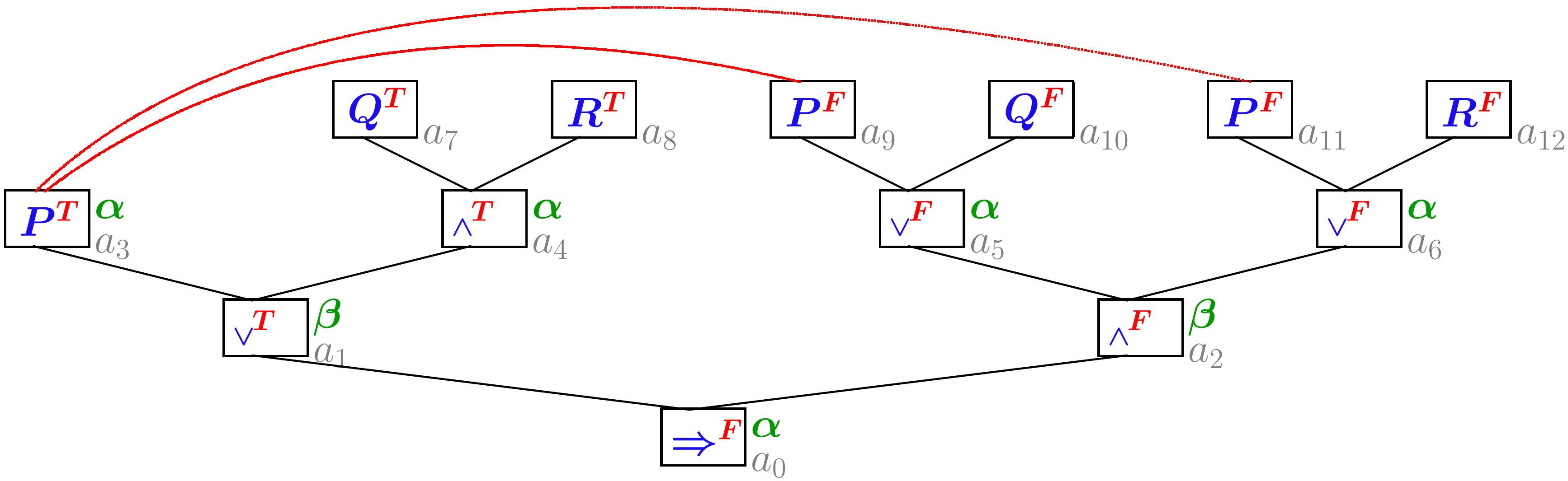


Parsen der Formel erzeugt Formelbaum

- **Zuweisung von Polaritäten:** T $\hat{=}$ Hypothese, F $\hat{=}$ Konklusion
- **Bestimmung des Typs:** α $\hat{=}$ linear, β $\hat{=}$ Verzweigung
- **Zuweisung von Polaritäten an Unterformeln**
- **Bestimmung des Typs der Unterformeln**
- **Erzeuge Konnektionen zwischen komplementären Literalen**

Matrixmethoden: annotierte Formelbäume

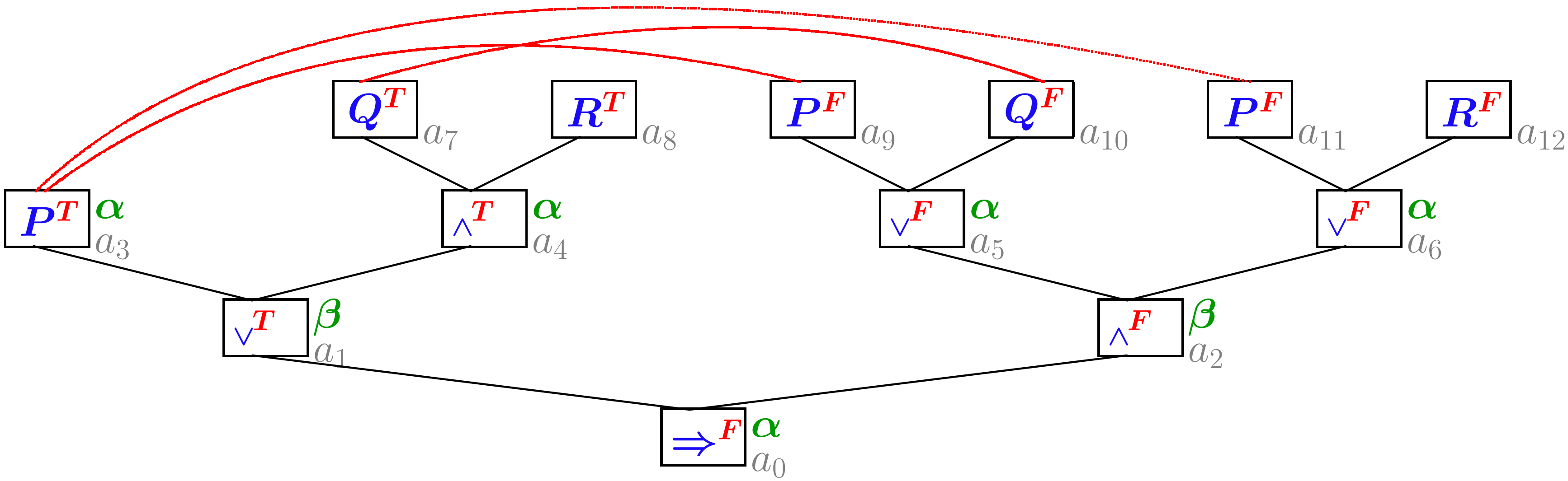
$$(P \vee (Q \wedge R)) \Rightarrow ((P \vee Q) \wedge (P \vee R))$$



Parsen der Formel erzeugt Formelbaum

- **Zuweisung von Polaritäten:** $T \;\hat{=}$ Hypothese, $F \;\hat{=}$ Konklusion
- **Bestimmung des Typs:** $\alpha \;\hat{=}$ linear, $\beta \;\hat{=}$ Verzweigung
- **Zuweisung von Polaritäten an Unterformeln**
- **Bestimmung des Typs der Unterformeln**
- **Erzeuge Konnektionen zwischen komplementären Literalen**

$$(P \lor (Q \land R)) \Rightarrow ((P \lor Q) \land (P \lor R))$$

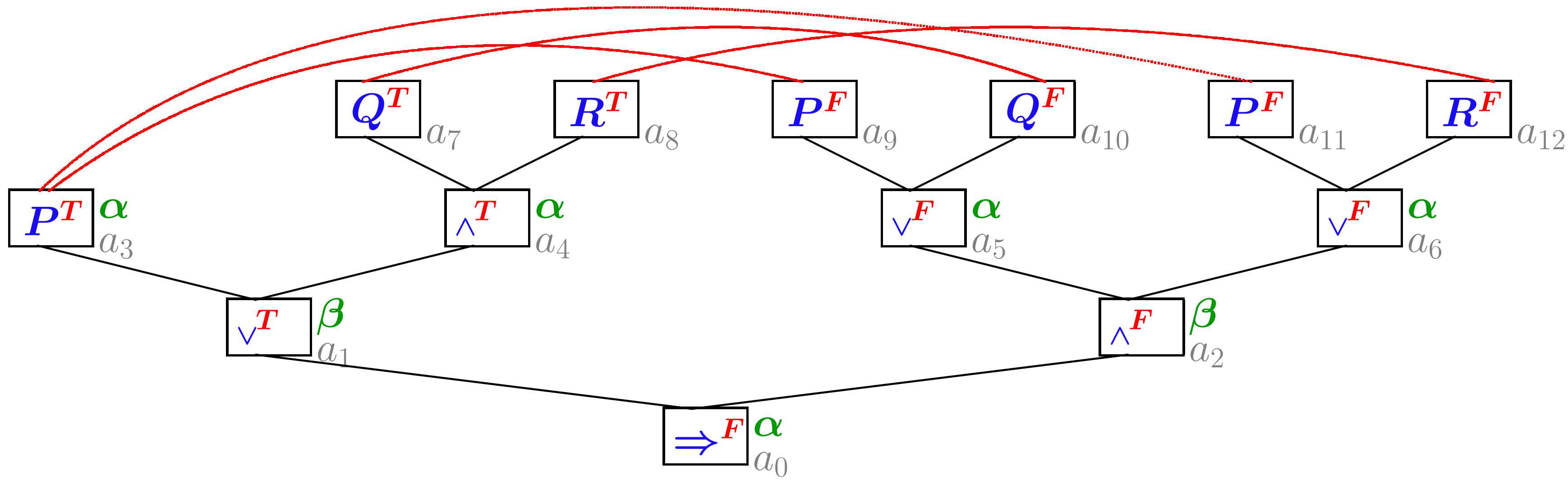


Parsen der Formel erzeugt Formelbaum

- **Zuweisung von Polaritäten:** T ≙ Hypothese, F ≙ Konklusion
- **Bestimmung des Typs:** α ≙ linear, β ≙ Verzweigung
- **Zuweisung von Polaritäten an Unterformeln**
- **Bestimmung des Typs der Unterformeln**
- **Erzeuge Konnektionen zwischen komplementären Literalen**

MATRIXMETHODEN: ANNOTIERTE FORMELBÄUME

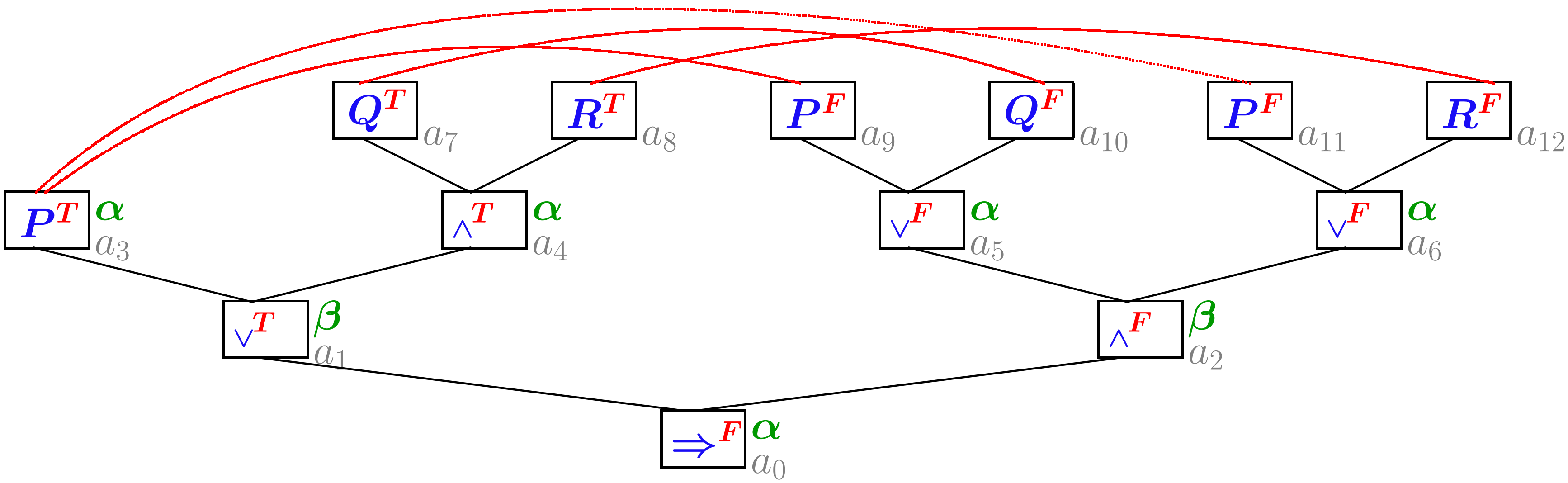
$$(P \vee (Q \wedge R)) \Rightarrow ((P \vee Q) \wedge (P \vee R))$$



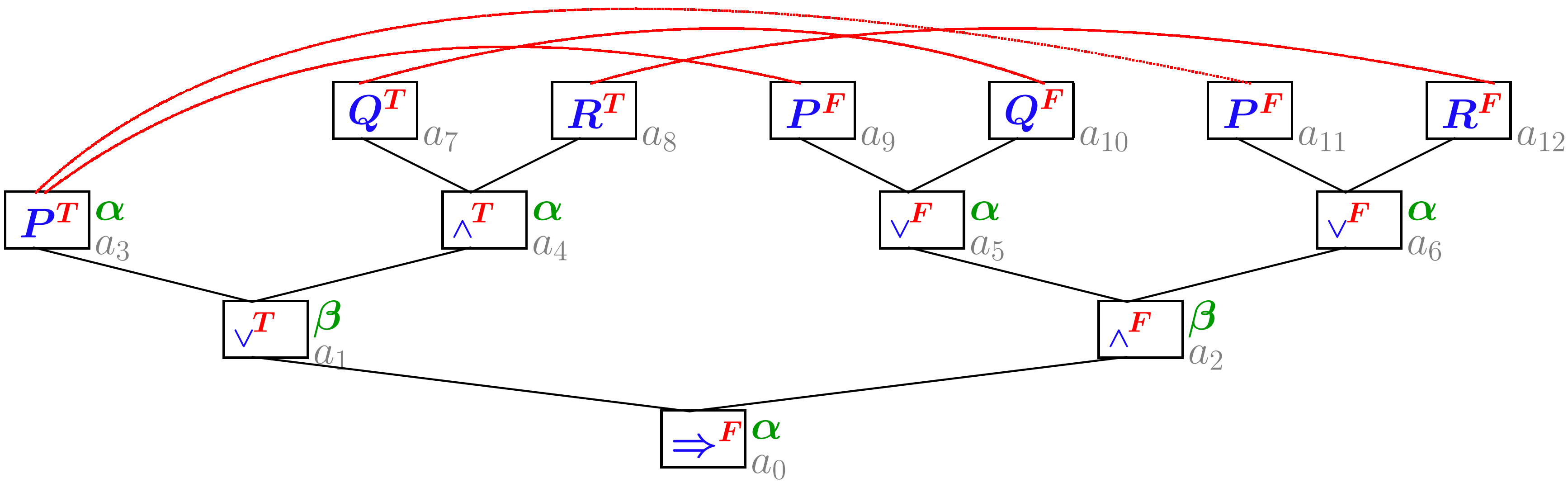
Parsen der Formel erzeugt Formelbaum

- **Zuweisung von Polaritäten:** $T \;\hat{=}\;$ Hypothese, $F \;\hat{=}\;$ Konklusion
- **Bestimmung des Typs:** $\alpha \;\hat{=}\;$ linear, $\beta \;\hat{=}\;$ Verzweigung
- **Zuweisung von Polaritäten an Unterformeln**
- **Bestimmung des Typs der Unterformeln**
- **Erzeuge Konnektionen zwischen komplementären Literalen**

Matrixmethoden: Analyse der Pfade

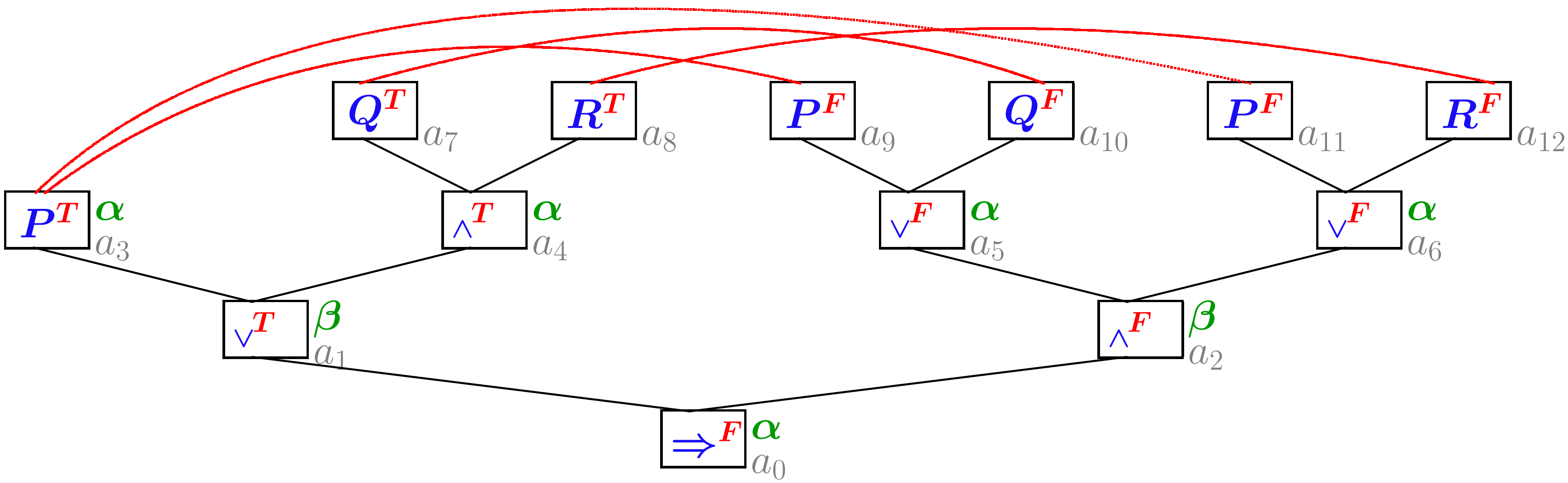


Matrixmethoden: Analyse der Pfade



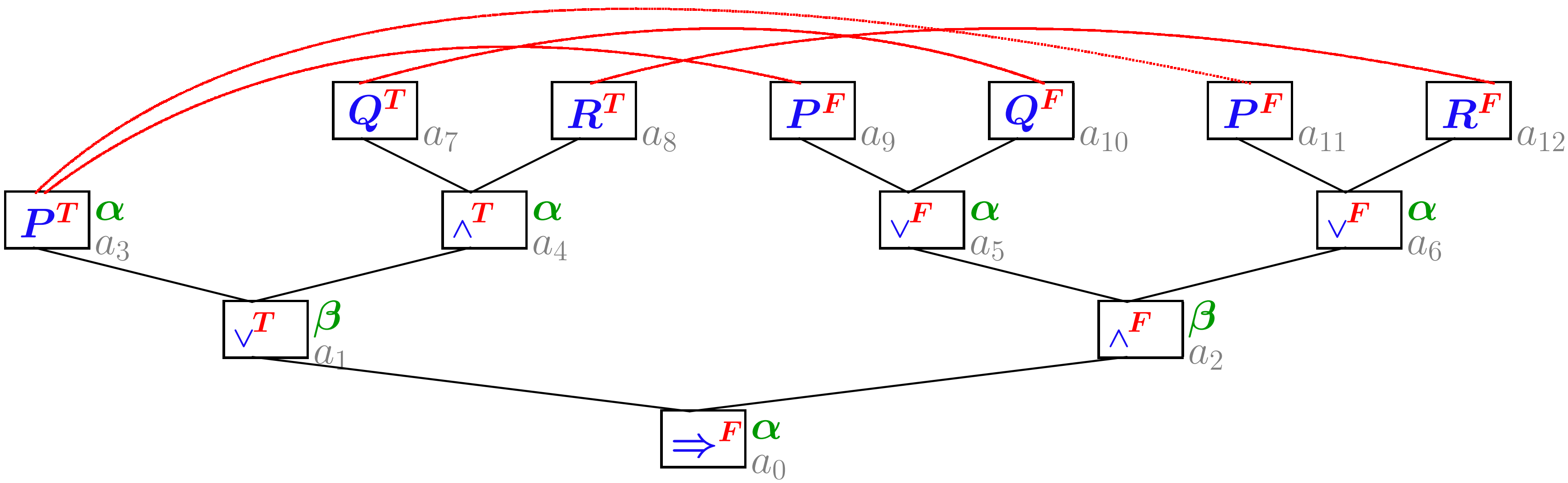
- **4 atomare Pfade** $\quad a_3a_9a_{10},\ a_3a_{11}a_{12},\ a_7a_8a_9a_{10},\ a_7a_8a_{11}a_{12}$

Matrixmethoden: Analyse der Pfade



- **4 atomare Pfade** $a_3a_9a_{10}$, $a_3a_{11}a_{12}$, $a_7a_8a_9a_{10}$, $a_7a_8a_{11}a_{12}$
- **Alle Pfade enthalten komplementäre Literale**

  Formel $(P \vee (Q \wedge R)) \Rightarrow ((P \vee Q) \wedge (P \vee R))$ **ist gültig**



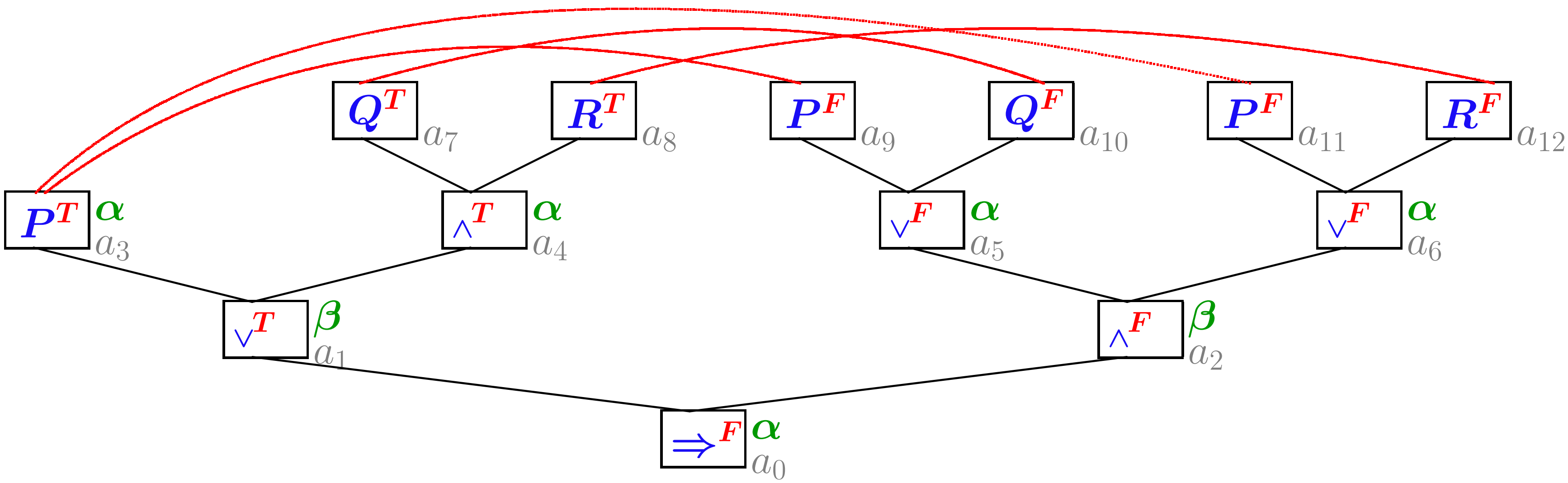
- **4 atomare Pfade** $a_3a_9a_{10}$, $a_3a_{11}a_{12}$, $a_7a_8a_9a_{10}$, $a_7a_8a_{11}a_{12}$
- **Alle Pfade enthalten komplementäre Literale**

  Formel $(P \vee (Q \wedge R)) \Rightarrow ((P \vee Q) \wedge (P \vee R))$ **ist gültig**

- **Zweidimensionale Repräsentation**

$$\begin{bmatrix} \begin{bmatrix} Q^T & R^T \end{bmatrix} & \begin{bmatrix} P^F & Q^F \end{bmatrix} \\ P^T & \begin{bmatrix} P^F & R^F \end{bmatrix} \end{bmatrix}$$

Matrixmethoden: Analyse der Pfade



- **4 atomare Pfade** $\quad a_3a_9a_{10},\ a_3a_{11}a_{12},\ a_7a_8a_9a_{10},\ a_7a_8a_{11}a_{12}$
- **Alle Pfade enthalten komplementäre Literale**

  Formel $(P \vee (Q \wedge R)) \Rightarrow ((P \vee Q) \wedge (P \vee R))$ **ist gültig**

- **Zweidimensionale Repräsentation**

  Mit Konnektionen

$$\begin{bmatrix} \begin{bmatrix} Q^T & R^T \end{bmatrix} & \begin{bmatrix} P^F & Q^F \end{bmatrix} \\ P^T & \begin{bmatrix} P^F & R^F \end{bmatrix} \end{bmatrix}$$

Matrixmethoden: zusätzliche Aspekte

- **Pfadüberprüfung folgt Konnektionen**
  - Frühzeitiges Abschneiden zu prüfender Pfade
  - Verringert Anzahl notwendiger Überprüfungen

$$\begin{bmatrix} \begin{bmatrix} Q^T & R^T \end{bmatrix} & \begin{bmatrix} P^F & Q^F \end{bmatrix} \\ P^T & \begin{bmatrix} P^F & R^F \end{bmatrix} \end{bmatrix}$$

Matrixmethoden: zusätzliche Aspekte

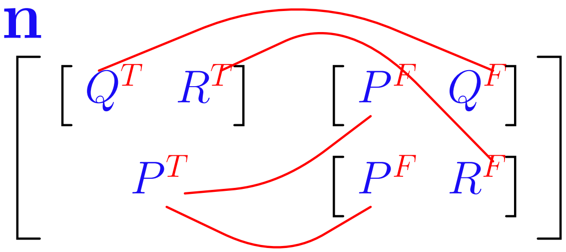
- **Pfadüberprüfung folgt Konnektionen**
  - Frühzeitiges Abschneiden zu prüfender Pfade
  - Verringert Anzahl notwendiger Überprüfungen

$$\begin{bmatrix} \begin{bmatrix} Q^T & R^T \end{bmatrix} & \begin{bmatrix} P^F & Q^F \end{bmatrix} \\ P^T & \begin{bmatrix} P^F & R^F \end{bmatrix} \end{bmatrix}$$

- **Logik erster Stufe braucht Term-Unifikation**
  - Variablen von γ-Knoten können instantiiert werden
  - Variablen von δ-Knoten gelten als Konstante
  - Standard-Algorithmen von Robinson oder Martelli-Montanari

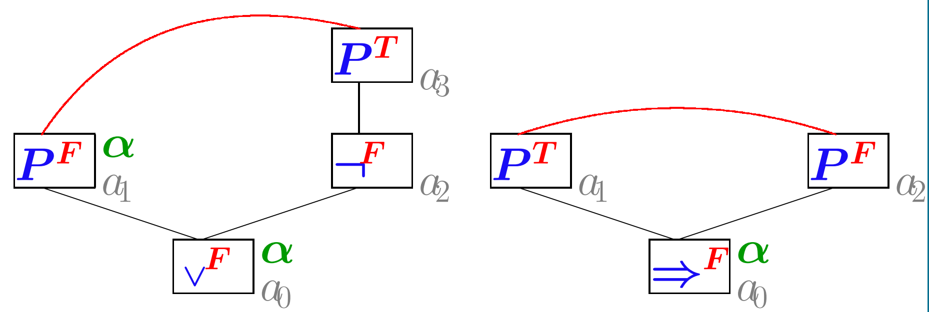
Matrixmethoden: zusätzliche Aspekte

- **Pfadüberprüfung folgt Konnektionen**
  – Frühzeitiges Abschneiden zu prüfender Pfade
  – Verringert Anzahl notwendiger Überprüfungen



- **Logik erster Stufe braucht Term-Unifikation**
  – Variablen von γ-Knoten können instantiiert werden
  – Variablen von δ-Knoten gelten als Konstante
  – Standard-Algorithmen von Robinson oder Martelli-Montanari

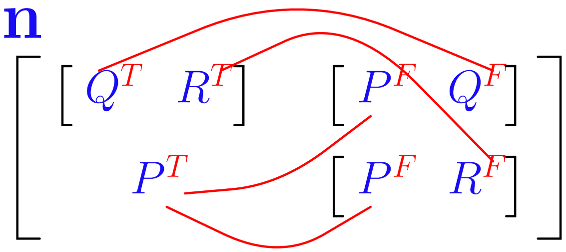
- **Konstruktive Logik braucht zusätzliche Methoden**
  – Unterscheide $P \vee \neg P$ von $P \Rightarrow P$
  – Regeln für $\Rightarrow$, $\neg$, $\forall$ sind irreversibel
  – Bestimme Reihenfolge der $\Rightarrow$, $\neg$, $\forall$
  – Hilfsmittel: Präfix(String)-Unifikation



$P \vee \neg P$ $\qquad$ $P \Rightarrow P$

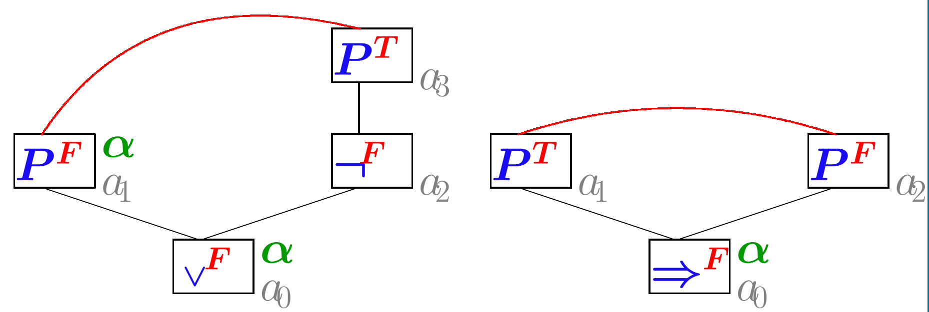
Matrixmethoden: zusätzliche Aspekte

- **Pfadüberprüfung folgt Konnektionen**
  - Frühzeitiges Abschneiden zu prüfender Pfade
  - Verringert Anzahl notwendiger Überprüfungen



- **Logik erster Stufe braucht Term-Unifikation**
  - Variablen von γ-Knoten können instantiiert werden
  - Variablen von δ-Knoten gelten als Konstante
  - Standard-Algorithmen von Robinson oder Martelli-Montanari

- **Konstruktive Logik braucht zusätzliche Methoden**
  - Unterscheide $P \vee \neg P$ von $P \Rightarrow P$
  - Regeln für $\Rightarrow$, $\neg$, $\forall$ sind irreversibel
  - Bestimme Reihenfolge der $\Rightarrow$, $\neg$, $\forall$
  - Hilfsmittel: Präfix(String)-Unifikation



$\boldsymbol{P \vee \neg P}$ $\boldsymbol{P \Rightarrow P}$

$\Downarrow$

Thema für separate Lehrveranstaltung

JProver: Integration von Matrixmethoden in Nuprl

Formel

$$\neg A \vee \neg B \Rightarrow \neg B \vee \neg A$$

JProver: Integration von Matrixmethoden in Nuprl

Formel

$\neg A \vee \neg B \Rightarrow \neg B \vee \neg A$

Annotierung

Typen, Polaritäten, Präfixe

JPROVER: INTEGRATION VON MATRIXMETHODEN IN Nuprl

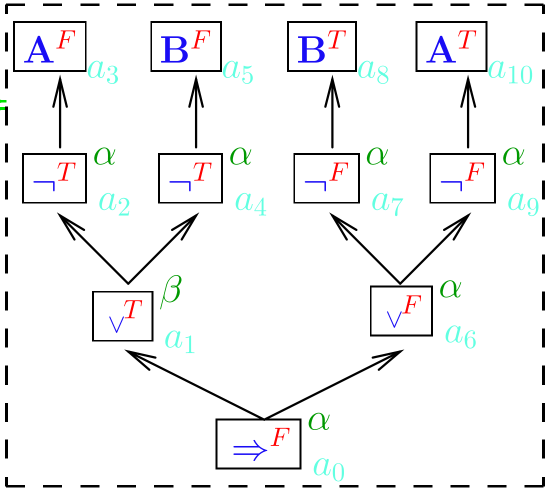
Formel

$\neg A \vee \neg B \Rightarrow \neg B \vee \neg A$

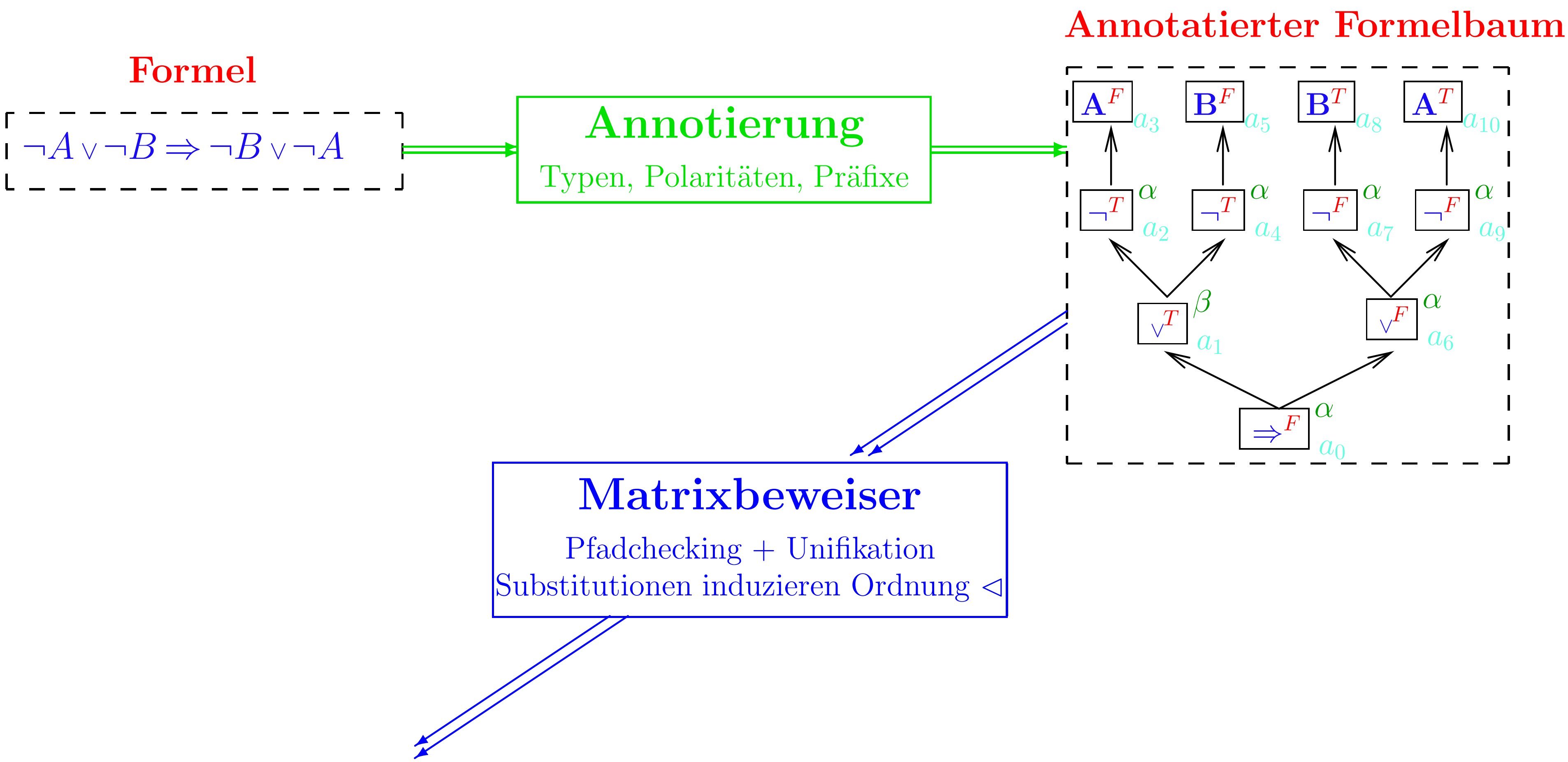
Annotierung

Typen, Polaritäten, Präfixe

Annotatierter Formelbaum



JProver: Integration von Matrixmethoden in Nuprl



JProver: Integration von Matrixmethoden in Nuprl

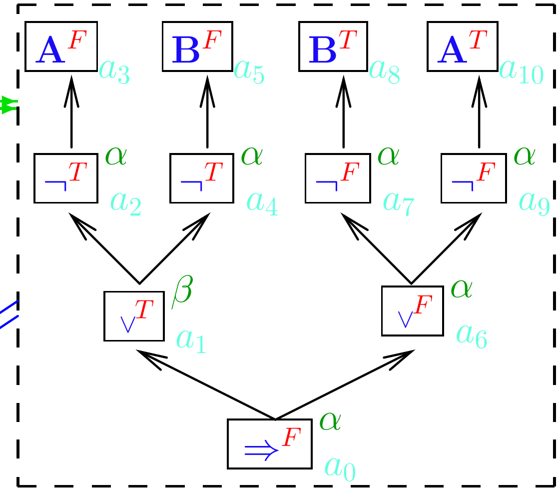
Formel

$\neg A \vee \neg B \Rightarrow \neg B \vee \neg A$

Annotierung

Typen, Polaritäten, Präfixe

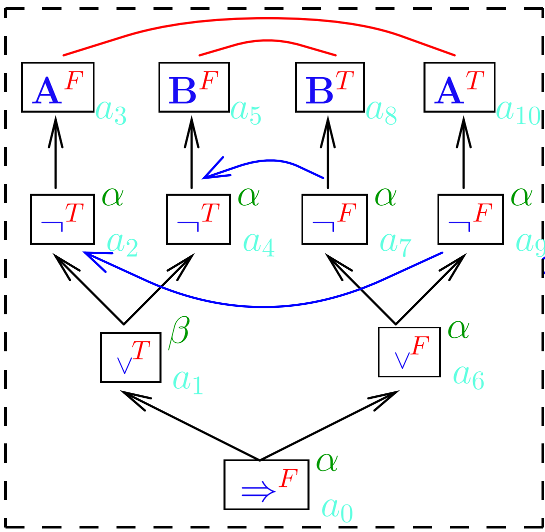
Annotatierter Formelbaum



Matrixbeweiser

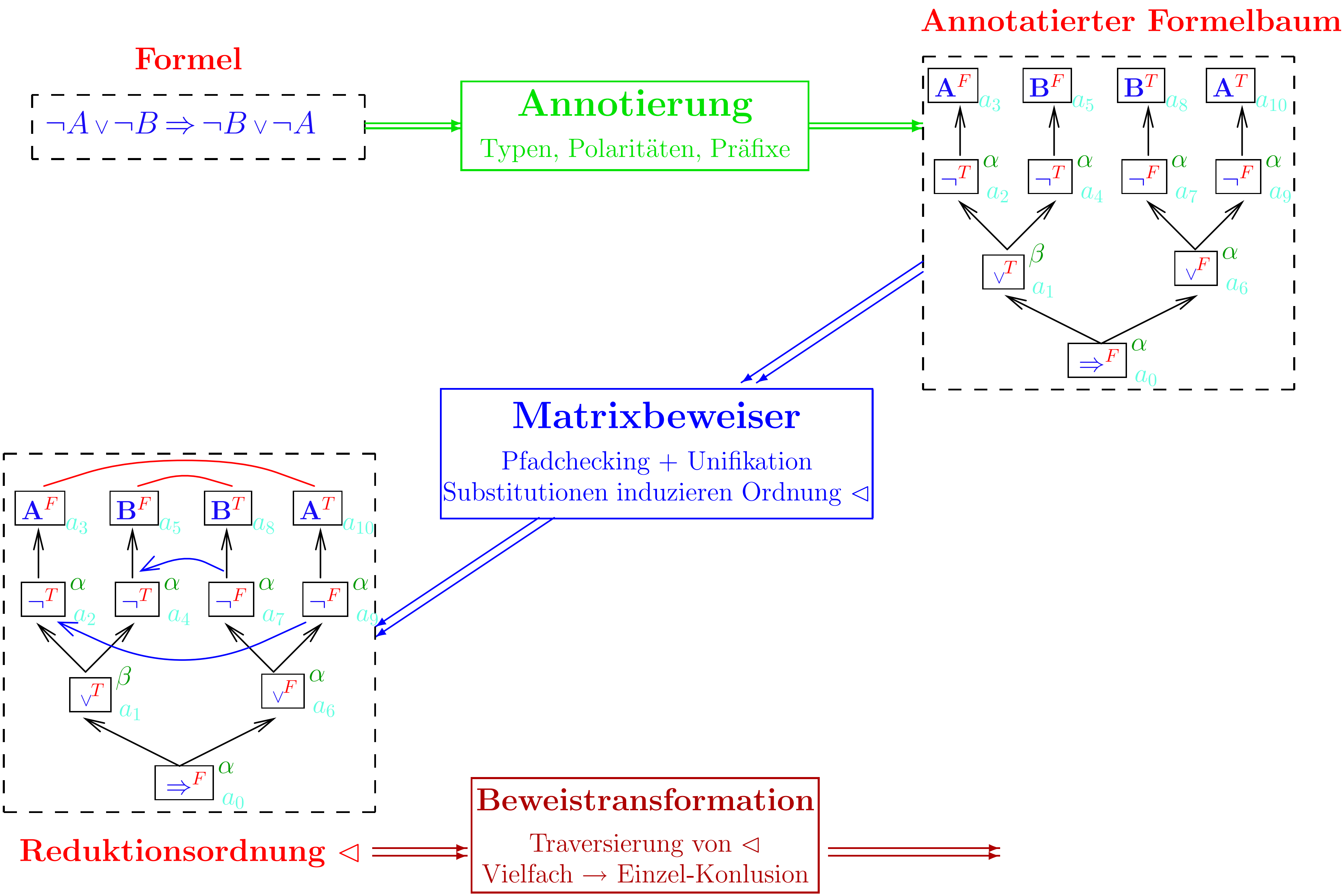
Pfadchecking + Unifikation

Substitutionen induzieren Ordnung $\lhd$

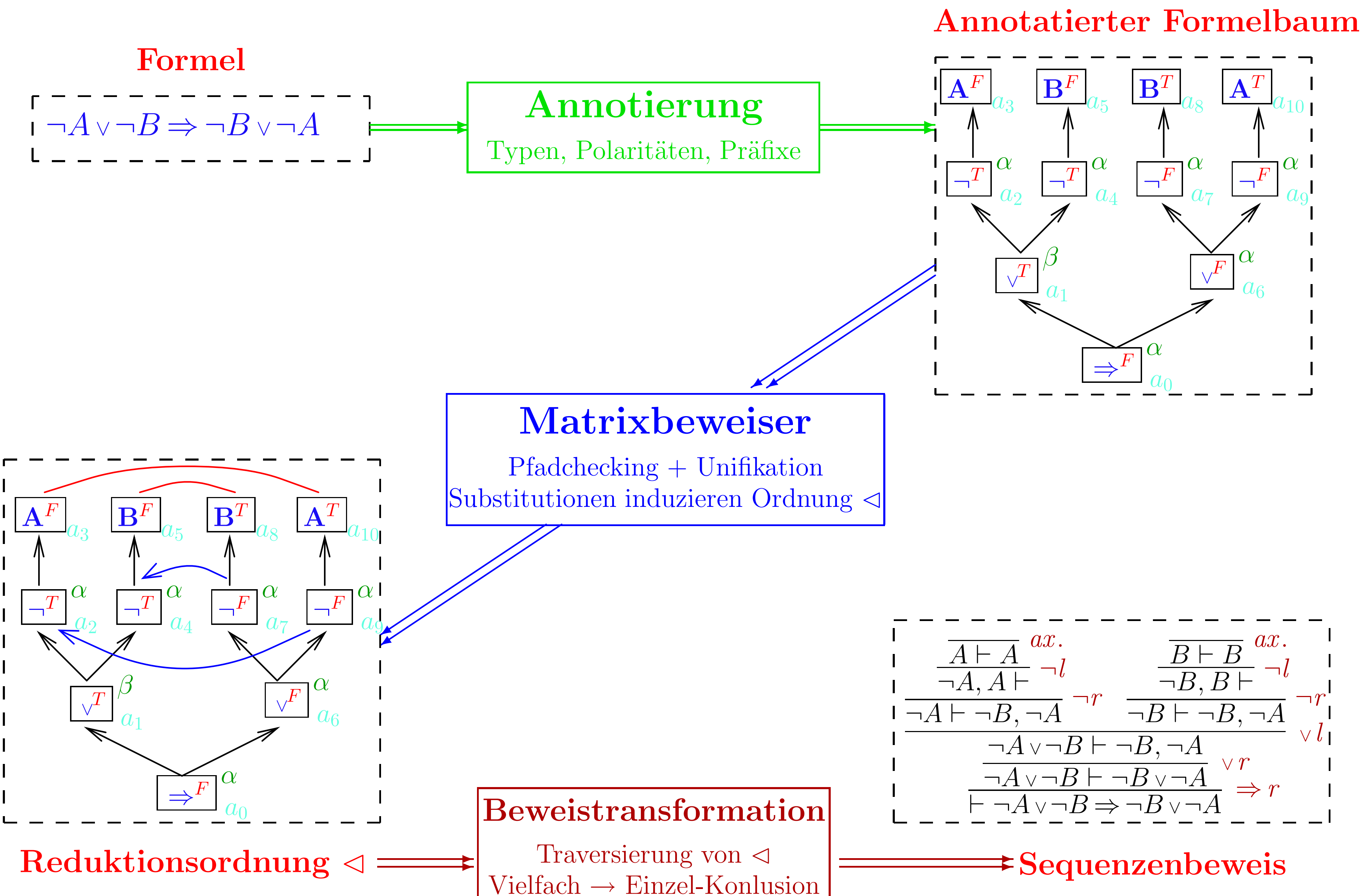


Reduktionsordnung $\lhd$

JProver: Integration von Matrixmethoden in Nuprl



JProver: Integration von Matrixmethoden in Nuprl



JProver: der automatische Beweiser

- **Beweissuche**
  - Matrixbeweiser für intuitionistische Logik erster Stufe (Kreitz & Otten 1999)
    (Konnektionsgetriebene Pfadüberprüfung + Termunifikation)
  - Zusätzliche Stringunifikation für konstruktive Beweise (Otten & Kreitz 1996)
  - Substitutionen und Formelbaum induzieren Reduktionsordnung

JProver: der automatische Beweiser

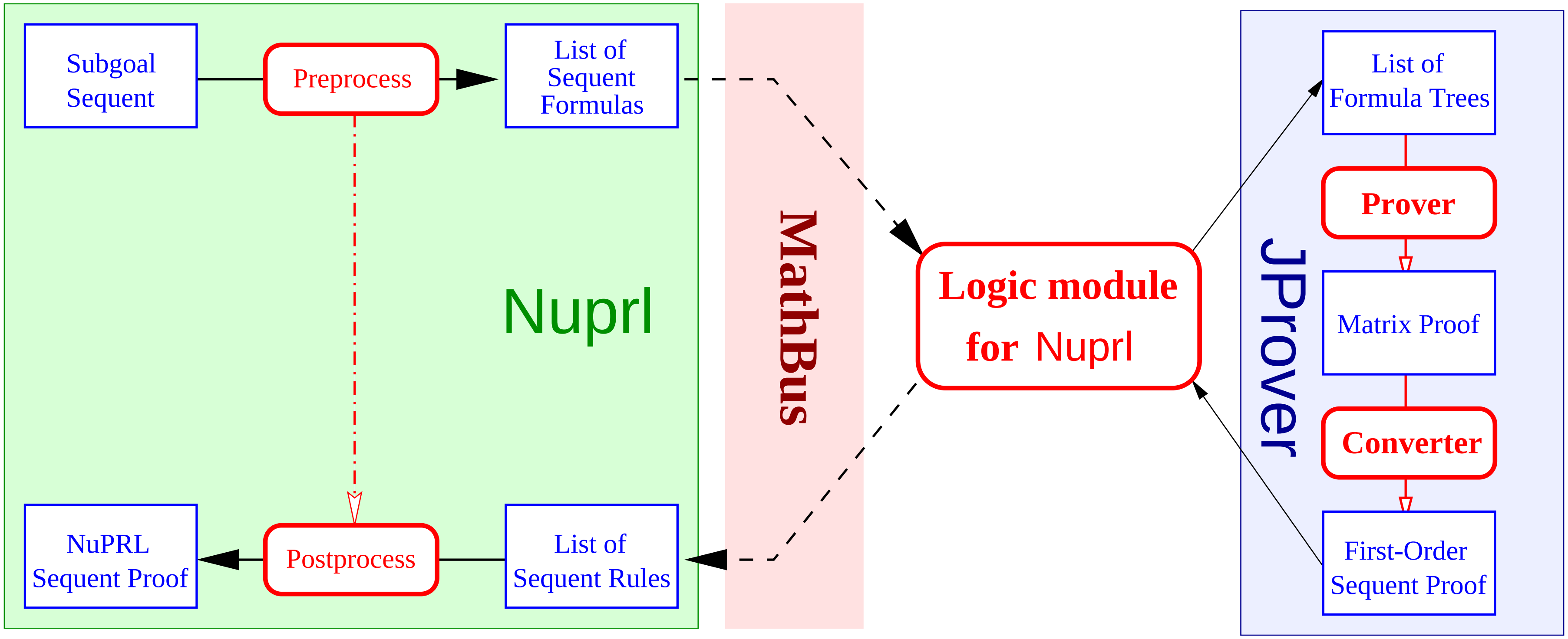
- **Beweissuche**
  – Matrixbeweiser für intuitionistische Logik erster Stufe (Kreitz & Otten 1999)
  (Konnektionsgetriebene Pfadüberprüfung + Termunifikation)
  – Zusätzliche Stringunifikation für konstruktive Beweise (Otten & Kreitz 1996)
  – Substitutionen und Formelbaum induzieren Reduktionsordnung

- **Beweistransformation**
  – Extrahiert Sequenzenbeweis aus Matrixbeweis (Kreitz & Schmitt 2000)
  – Traversiert Reduktionsordnung ohne Suche (Schmitt 2000)
  – Handhabt Sequenzenkalküle mit mehreren/ einer Konlusion (Egly & Schmitt 1999)

JProver: der automatische Beweiser

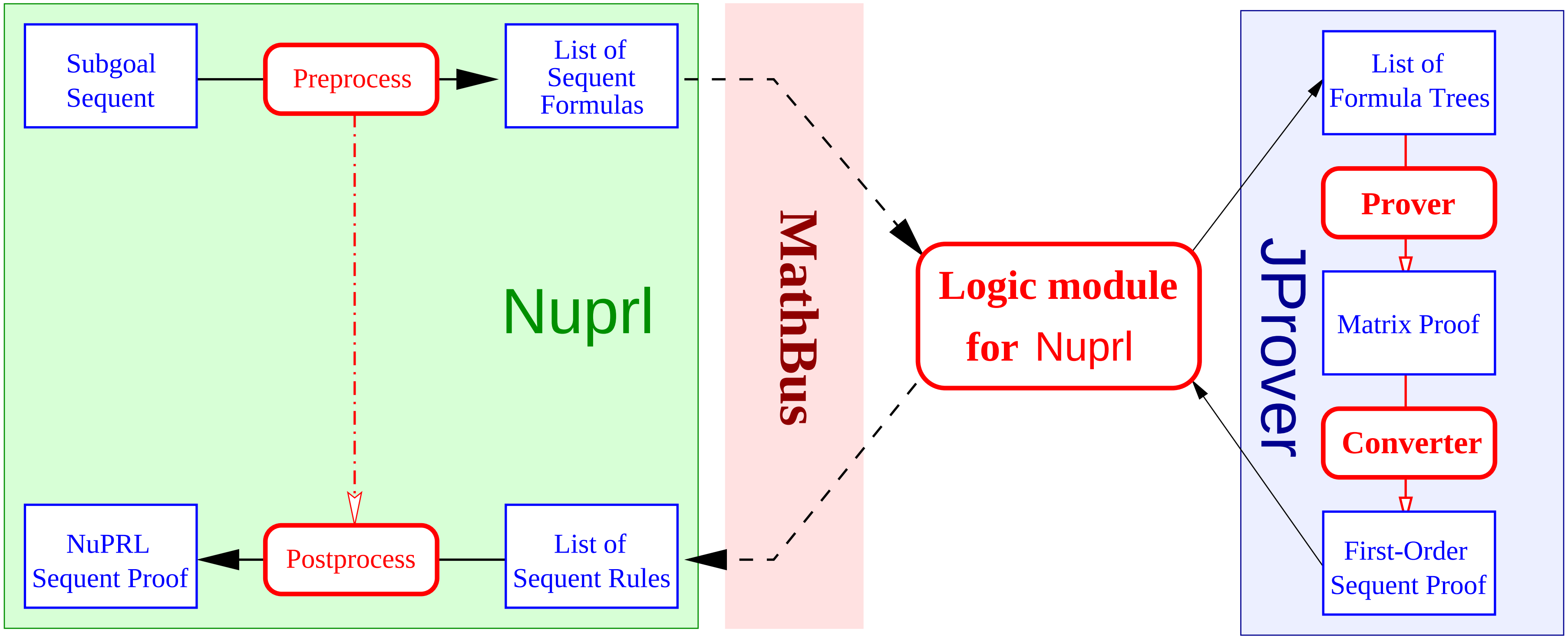
- **Beweissuche**
  - Matrixbeweiser für intuitionistische Logik erster Stufe (Kreitz & Otten 1999)
    (Konnektionsgetriebene Pfadüberprüfung + Termunifikation)
  - Zusätzliche Stringunifikation für konstruktive Beweise (Otten & Kreitz 1996)
  - Substitutionen und Formelbaum induzieren Reduktionsordnung
- **Beweistransformation**
  - Extrahiert Sequenzenbeweis aus Matrixbeweis (Kreitz & Schmitt 2000)
  - Traversiert Reduktionsordnung ohne Suche (Schmitt 2000)
  - Handhabt Sequenzenkalküle mit mehreren/ einer Konlusion (Egly & Schmitt 1999)
- **Implementierung** (Schmitt et. al 2001)
  - Stand-alone Beweiser in OCaml
  - Einbettung in MetaPRL-Umgebung liefert Basisfunktionalitäten
    (Datentypen für Terme, Termunifikation, Modul System)

JProver: Anbindung an Nuprl



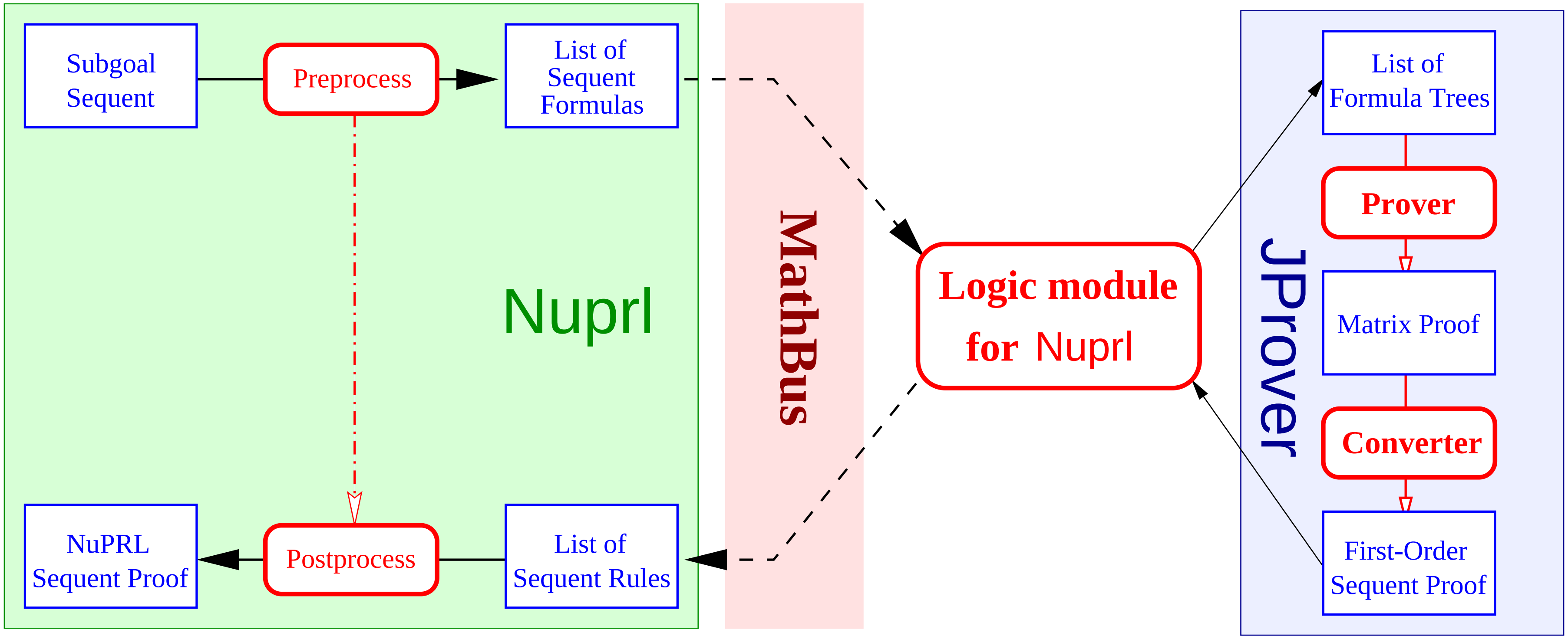
- Präprozessor für Nuprl Sequenzen und semantische Unterschiede

JProver: Anbindung an Nuprl



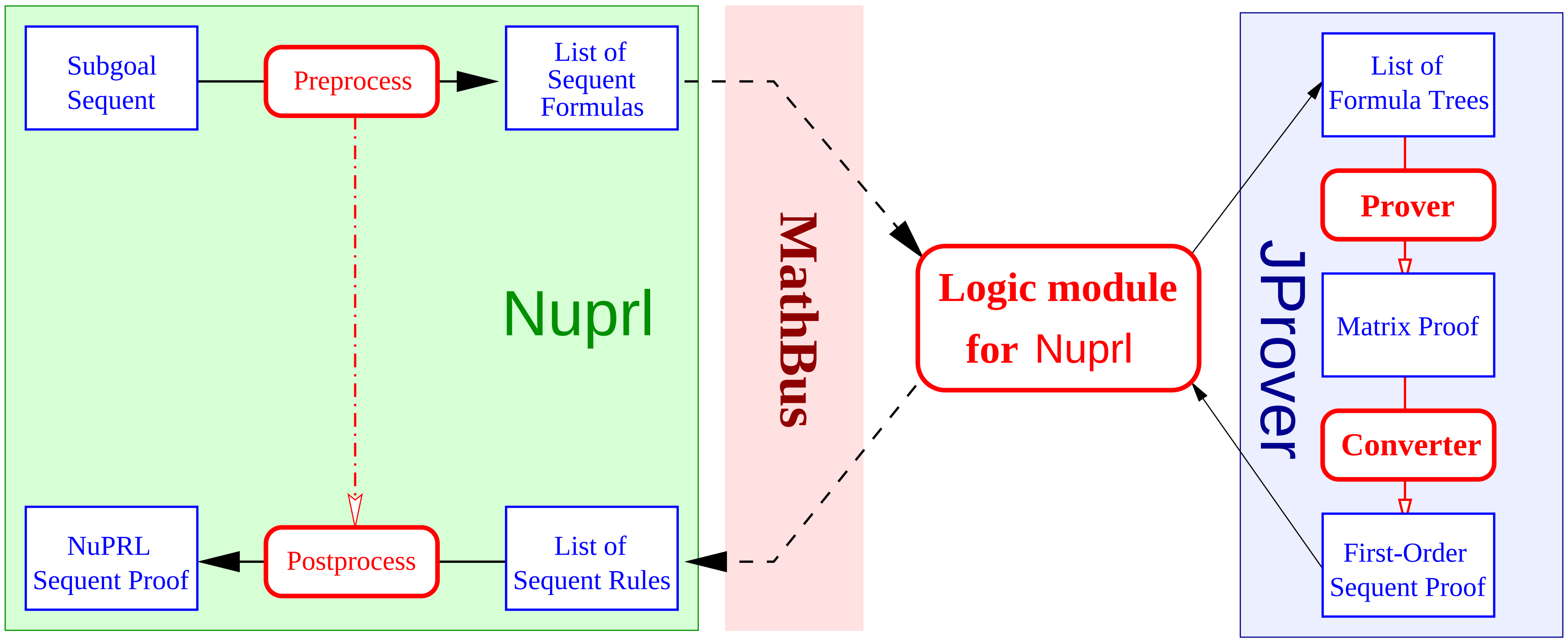
- Präprozessor für **Nuprl** Sequenzen und semantische Unterschiede
- Kommunikation von Termen im MathBus Format über INET socket

JProver: Anbindung an Nuprl



- Präprozessor für **Nuprl** Sequenzen und semantische Unterschiede
- Kommunikation von Termen im MathBus Format über INET socket
- `JLogic` Modul: extrahiert semantische Information aus Termen und konvertiert Sequenzenbeweis in das Format von **Nuprl**

JProver: ANBINDUNG AN Nuprl



- Präprozessor für Nuprl Sequenzen und semantische Unterschiede
- Kommunikation von Termen im MathBus Format über INET socket
- JLogic Modul: extrahiert semantische Information aus Termen und konvertiert Sequenzenbeweis in das Format von Nuprl
- Postprozessor baut Nuprl Beweisbaum für Ausgangssequenz

Logische Integration in Nuprl

- **Logikmodul: Komponenten**
  - OCaml code für Kommunikation mit interaktivem Beweiser
  - JLogic Modul zur Darstellung Nuprl Logik

Logische Integration in Nuprl

- **Logikmodul: Komponenten**
  - OCaml code für Kommunikation mit interaktivem Beweiser
  - `JLogic` Modul zur Darstellung Nuprl Logik

- **Das `JLogic` Modul**
  - Beschreibt Terme, welche Nuprl's logische Konnektive implementieren
  - Liefert Operationen zum Zugriff auf Teilterme
  - Decodiert Sequenzen, die in MathBus Format ankommen
  - Codiert JProver's Sequenzenbeweis ins MathBus Format

Logische Integration in Nuprl

- **Logikmodul: Komponenten**
  - OCaml code für Kommunikation mit interaktivem Beweiser
  - `JLogic` Modul zur Darstellung Nuprl Logik

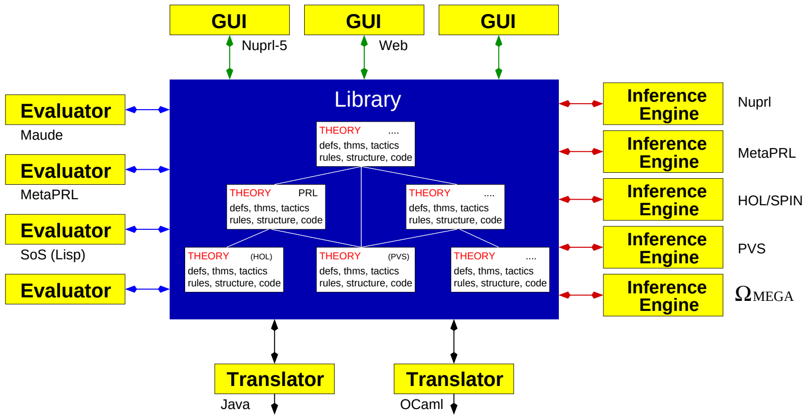
- **Das `JLogic` Modul**
  - Beschreibt Terme, welche Nuprl's logische Konnektive implementieren
  - Liefert Operationen zum Zugriff auf Teilterme
  - Decodiert Sequenzen, die in MathBus Format ankommen
  - Codiert JProver's Sequenzenbeweis ins MathBus Format

```
module Nuprl_JLogic =
struct
let is_all_term = nuprl_is_all_term
let dest_all = nuprl_dest_all
let is_exists_term = nuprl_is_exists_term
let dest_exists = nuprl_dest_exists
let is_and_term = nuprl_is_and_term
let dest_and = nuprl_dest_and
let is_or_term = nuprl_is_or_term
let dest_or = nuprl_dest_or
let is_implies_term = nuprl_is_implies_term
let dest_implies = nuprl_dest_implies
let is_not_term = nuprl_is_not_term
let dest_not = nuprl_dest_not

type inference = '(string*term*term) list
let empty_inf = []
let append_inf inf t1 t2 r =
    ((Jall.ruletable r), t1, t2) :: inf
end
```

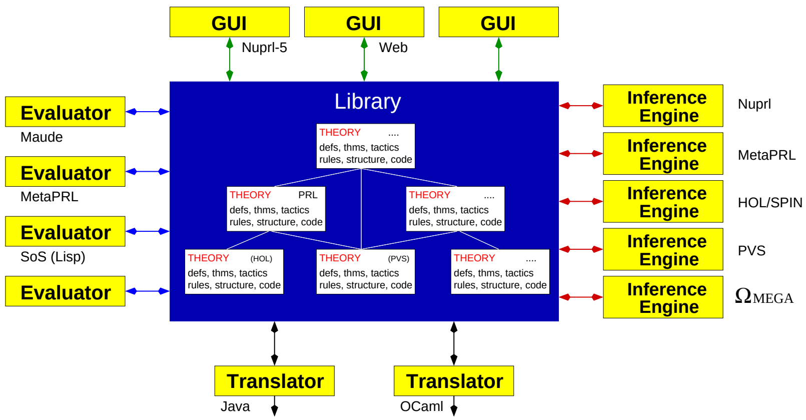
Der Weg zur logischen Wissensbank ...

- **Verbinde externe Systeme**



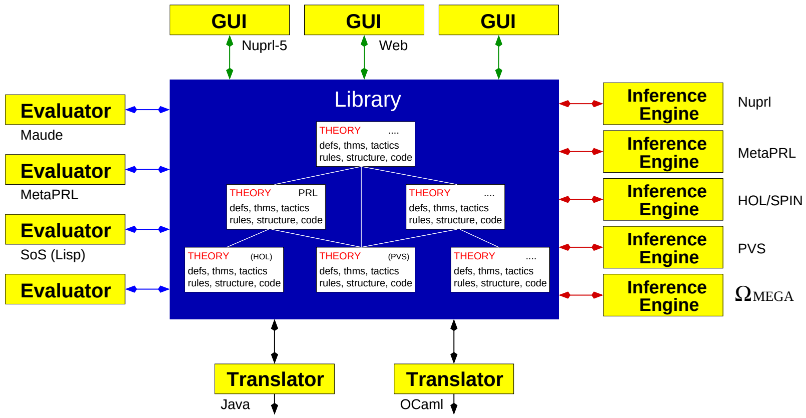
Der Weg zur logischen Wissensbank ...

- **Verbinde externe Systeme**
  - Beweissysteme: PVS, HOL, ...



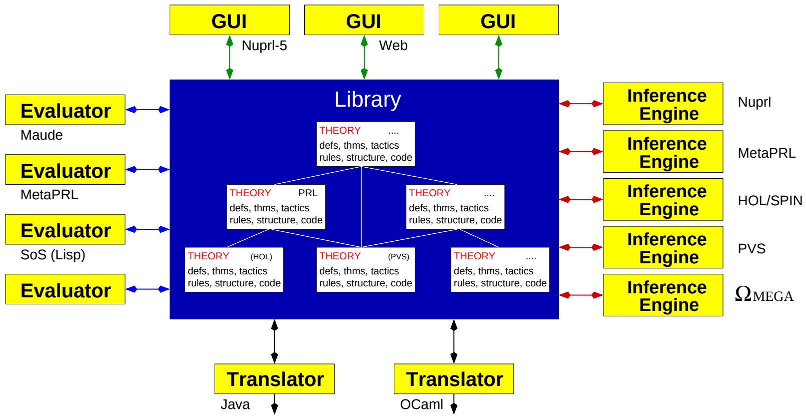
Der Weg zur logischen Wissensbank . . .

- **Verbinde externe Systeme**
  - Beweissysteme: PVS, HOL, . . .
  - Browser (ASCII, web,. . .) und Editoren (strukturiert, Emacs-mode,. . .)



Der Weg zur logischen Wissensbank ...

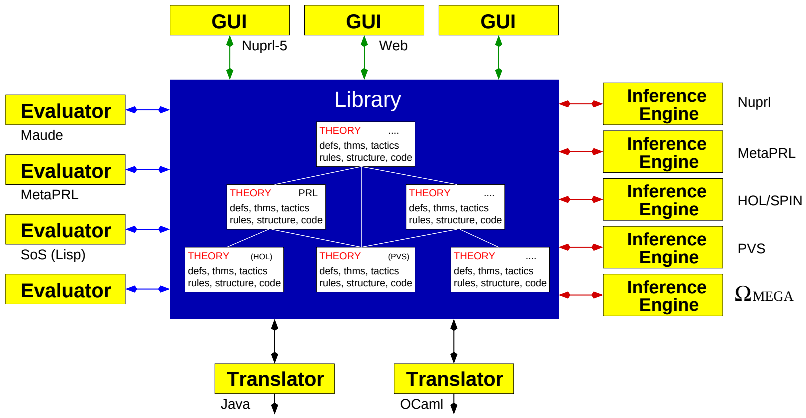
- **Verbinde externe Systeme**
  - Beweissysteme: PVS, HOL, ...
  - Browser (ASCII, web,...) und Editoren (strukturiert, Emacs-mode,...)



- **Ergänze neue Kapazitäten**
  - Archivierung (Dokumentation & Zertifizierung, Versionskontrolle)

Der Weg zur logischen Wissensbank . . .

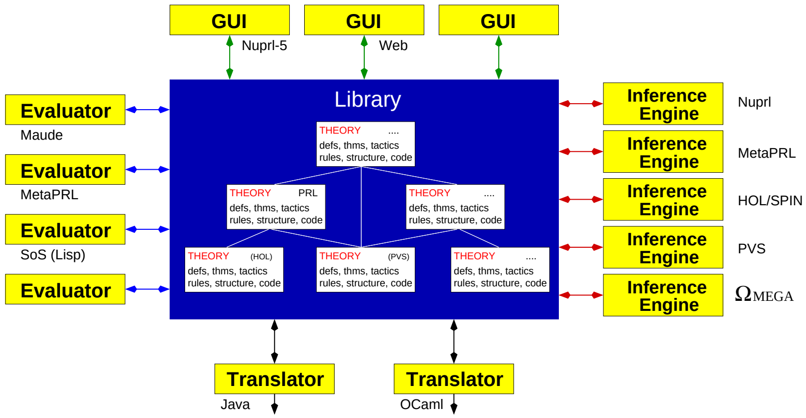
- **Verbinde externe Systeme**
  - Beweissysteme: PVS, HOL, . . .
  - Browser (ASCII, web,. . .) und Editoren (strukturiert, Emacs-mode,. . .)



- **Ergänze neue Kapazitäten**
  - Archivierung (Dokumentation & Zertifizierung, Versionskontrolle)
  - Einbettung des Inhalts externer Wissensbanken

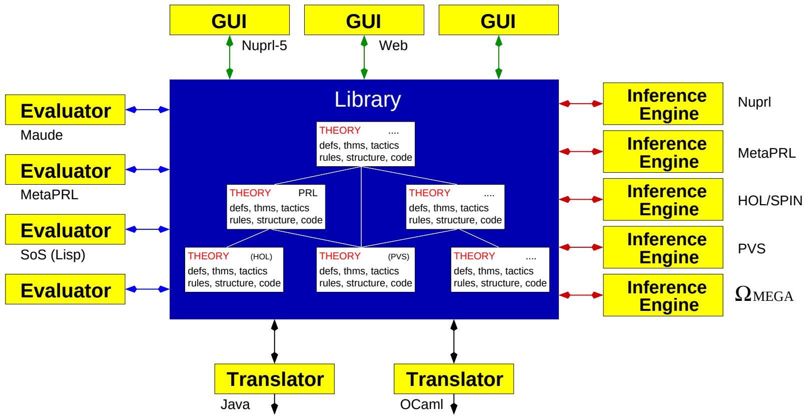
Der Weg zur logischen Wissensbank . . .

- **Verbinde externe Systeme**
  - Beweissysteme: PVS, HOL, . . .
  - Browser (ASCII, web,. . .) und Editoren (strukturiert, Emacs-mode,. . .)



- **Ergänze neue Kapazitäten**
  - Archivierung (Dokumentation & Zertifizierung, Versionskontrolle)
  - Einbettung des Inhalts externer Wissensbanken
  - Eine Vielfalt von Rechtfertigungen (verschiedene Vertrauensstufen)

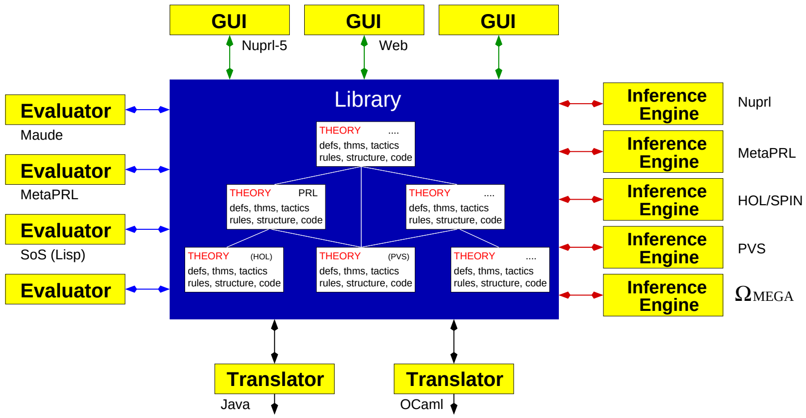
Der Weg zur logischen Wissensbank ...



- **Verbinde externe Systeme**
  - Beweissysteme: PVS, HOL, ...
  - Browser (ASCII, web,...) und Editoren (strukturiert, Emacs-mode,...)
- **Ergänze neue Kapazitäten**
  - Archivierung (Dokumentation & Zertifizierung, Versionskontrolle)
  - Einbettung des Inhalts externer Wissensbanken
  - Eine Vielfalt von Rechtfertigungen (verschiedene Vertrauensstufen)
  - Erzeugung formaler und textlicher Dokumente

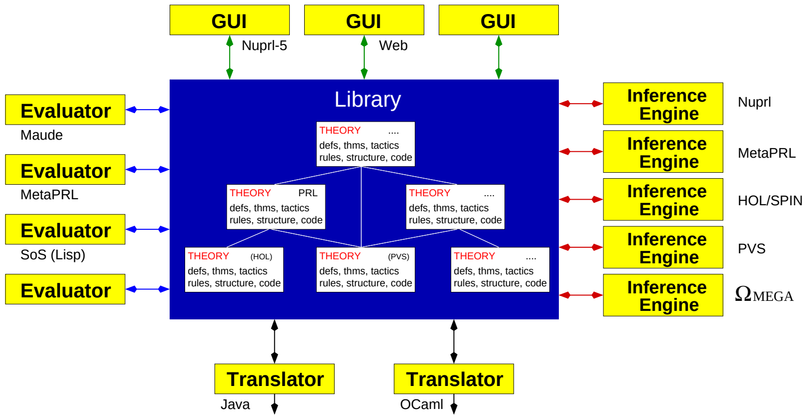
Der Weg zur logischen Wissensbank ...

- **Verbinde externe Systeme**
  - Beweissysteme: PVS, HOL, ...
  - Browser (ASCII, web,...) und Editoren (strukturiert, Emacs-mode,...)



- **Ergänze neue Kapazitäten**
  - Archivierung (Dokumentation & Zertifizierung, Versionskontrolle)
  - Einbettung des Inhalts externer Wissensbanken
  - Eine Vielfalt von Rechtfertigungen (verschiedene Vertrauensstufen)
  - Erzeugung formaler und textlicher Dokumente
  - Asynchrone und verteilte Operation

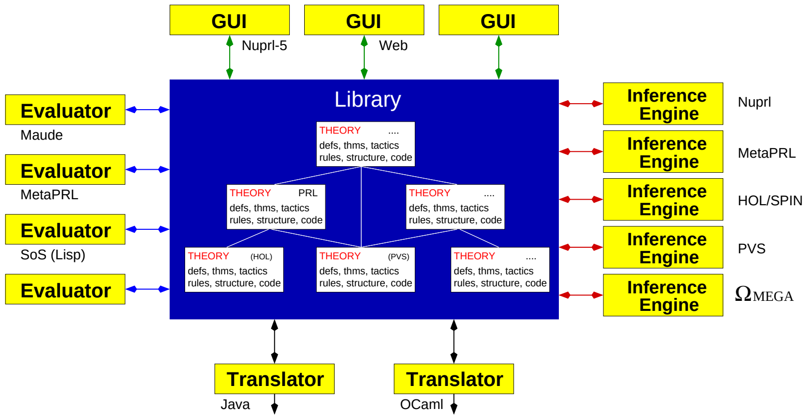
Der Weg zur logischen Wissensbank . . .



- **Verbinde externe Systeme**
  - Beweissysteme: PVS, HOL, . . .
  - Browser (ASCII, web,. . .) und Editoren (strukturiert, Emacs-mode,. . .)

- **Ergänze neue Kapazitäten**
  - Archivierung (Dokumentation & Zertifizierung, Versionskontrolle)
  - Einbettung des Inhalts externer Wissensbanken
  - Eine Vielfalt von Rechtfertigungen (verschiedene Vertrauensstufen)
  - Erzeugung formaler und textlicher Dokumente
  - Asynchrone und verteilte Operation
  - Meta-schließen (z.B.. über Bezüge zwischen verschiedenen Theorien)

Der Weg zur logischen Wissensbank ...



- **Verbinde externe Systeme**
  - Beweissysteme: PVS, HOL, ...
  - Browser (ASCII, web,...) und Editoren (strukturiert, Emacs-mode,...)
- **Ergänze neue Kapazitäten**
  - Archivierung (Dokumentation & Zertifizierung, Versionskontrolle)
  - Einbettung des Inhalts externer Wissensbanken
  - Eine Vielfalt von Rechtfertigungen (verschiedene Vertrauensstufen)
  - Erzeugung formaler und textlicher Dokumente
  - Asynchrone und verteilte Operation
  - Meta-schließen (z.B.. über Bezüge zwischen verschiedenen Theorien)

$\Downarrow$

Referenzumgebung für Entwicklung zuverlässiger Softwaree