Deduction

Deductive systems

Deductive databases

Parsing as deduction

Explaining deductive reasoning

DEDUCTIVE INFERENCING

COMPUTATION

=

DEDUCTION

+

CONTROL

Kowalski

LAYERS OF A DEDUCTION SYSTEM

Four layers connecting deduction and control

1. **Logic – syntax and semantics of expressions**
2. **Calculus – syntactic derivations on formulas**
3. **Representation – state of formulas and derivations**
4. **Control – strategies and heuristics for selection**

Most common examples

1.	**First-order predicate logic**		
2.	**Gentzen calculi**	**Resolution**	**Theory resolution**
3.	**Tableau**	**Matrix**	**Clause graphs**
4.	**Special techniques**		

PROPERTIES OF CALCULI

Function

Syntactic **derivation to verify** ***semantic*** **validity (true under all interpretations)**

Structure

Set of ***axioms*** **(tautologies), minimal and** ***derivation rules***

Behavior

Forward chaining – deductive calculus

Backward chaining – test calculus

Assessing a calculus

Soundness **– all axioms and derivable formulas are valid**

Completeness **– all valid formulas are derivable (in a finite number of steps)**

Some fundamental insights

First-order predicate logic is ***not decidable*****, but** ***complete***

Higher-order predicate logic is ***not complete*** **(Gödel 1931)**

First-order predicate logic is the ***most expressive*** **and still** ***complete*** **logic (Lindström 1969)**

GENTZEN CALCULUS – NATURAL DEDUCTION

A positive deductive calculus

13 Derivation rules

$$\frac{F \quad G}{F \wedge G} \qquad \frac{F}{F \vee G} \quad \frac{G}{F \vee G} \qquad \frac{\begin{matrix}[F]\\ G\end{matrix}}{F \Rightarrow G} \qquad \frac{Fa}{\forall x Fx} \qquad \frac{Fa}{\exists x Fx} \qquad \frac{\begin{matrix}[F]\\ \square\end{matrix}}{\neg F}$$

And-Intro. **Or-Intro.** **Implication-Intro.** **All-Intro.** **Exist-Intro.** **Negation-Intro.**

$$\frac{F \wedge G}{F} \quad \frac{F \wedge G}{G} \qquad \frac{F \vee G \quad \begin{matrix}[F]\\ H\end{matrix} \quad \begin{matrix}[G]\\ H\end{matrix}}{H} \qquad \frac{F \quad F \Rightarrow G}{G} \qquad \frac{\forall x Fx}{Fa} \qquad \frac{\exists x Fx \quad \begin{matrix}[Fa]\\ H\end{matrix}}{H} \qquad \frac{F \quad \neg F}{\square} \quad \frac{\square}{F}$$

And-Elim. **Or-Elim.** **Implication-Elim.** **All-Elim.** **Exist-Elim.** **Negation-Elim.**

One axiom ($F \vee \neg F$) needed for obtaining completeness

RESOLUTION (Robinson 1965)

A negative test calculus

Uses formulas in form of clauses

(set of literals, implicitly disjoint)

- **One axiom, elementary contradiction**

  □ (empty clause)
- **One resolution rule**

	Simplest form:	*Resolution with substitution:*	
clause1:	$L, K_1, \ldots, K_n$	$L, K_1, \ldots, K_n$	
clause2:	$\neg L, M_1, \ldots, M_m$	$\neg L', M_1, \ldots, M_m$	$\sigma L = \sigma L'$
resolvent:	$K_1, \ldots, K_n, M_1, \ldots, M_m$	$\sigma K_1, \ldots, \sigma K_n, \sigma M_1, \ldots, \sigma M_m$	

Resolution has *refutation completeness*

RESOLUTION - AN EXAMPLE

A barber shaves a person if and only if

that person does not shave himself.

Formalization of the statement and normalization into disjunctive Normal form

shave(barber,x) ↔ ¬shave(x,x) transformed into:

(shave(barber,x) → ¬shave(x,x)) ^ (¬shave(x,x) → shave(barber,x))

(¬shave(barber,x) ∨ ¬shave(x,x)) ^ (shave(x,x) ∨ shave(barber,x))

Factorization prior to substitution:

σ = {x ← barber, y ← barber}

shave(x,x), shave(barber,x) ⊢ shave(barber,barber)

¬shave(barber,y), ¬shave(y,y) ⊢ ¬shave(barber,barber)

□

THEORY RESOLUTION

Extension of the resolution principle to a domain theory

No interpretation makes both L and ¬L true

- **in resolution only for *syntactic* complementarity**
- **extended to (multiple) *semantic* complementarity**

An example

clause1: a<b, K

clause2: b<c, M

clause3: c<a, N

---

resolvent: K, M, N

REPRESENTATION

Efficiency through reduction rules

- ***logical*** **simplifications**

  (tautologies replaced by true, contradictions by false, replacements similar to elimination rules in ND calculus)

- **simplifications with '*useless*' clauses**

  (e.g.: isolation rule – clause with 'unique' literal)

Design of reduction rules 'creative', needs justification

Consequences of representation

State transition instead of calculus

- ***Initial*** **states are sets of clauses**
- ***Intermediate*** **states are e.g., clauses graphs**
- ***Terminal*** **states – with or without empty clause**

CONTROL

Efficiency through syntactically-based strategies

- ***Restriction*** **strategies**
  (e.g., unit resolution – complete for Horn clauses, input resolution, set-of-support, …)

- ***Ordering*** **strategies**
  (e.g.: saturation level, with unit preference, fairness, unit resolution, …)

An example prover - OTTER

- **Elaborate implementation, indexing techniques**
- **User categorises clauses into (ordinary) clauses, axioms, and set of support, specifies options**
- **Makes resolution with axioms and 'best' clause from set-of-support**

DEDUCTIVE DATABASES

Components

- **Tables (*existensional* database)**
- **Rules (*intensional* database)**

Rules define intensional database on the base of the extensional one

Procedure

- **Elementary relations expressed in tables**
- **General, recursive relations expressed in rules**
- **Query evaluation through resolution derivation**

Benefit

- **Combines capabilities of relational databases and rule-based systems**
- **Strong reduction of redundancy and number of tables**

Problem: Control over chained application of rules

REPRESENTATION IN A RELATIONAL DATABASE

Example database relation

PARENT	
CNAME	PNAME
Smith John Jr	Smith John
Smith John Jr	Smith Mary
Rogers Charles	Rogers Linda
Rogers Linda	Jones David
Rogers Linda	Jones Mary
Smith Mary	Ford Albert
Cramer Steven	Cramer William

Relational calculus query to find the parent(s) of Charles Rogers

- **GET(X) : PARENT(Rogers Charles, X)**

REPRESENTATION WITH TWO TABLES

Example database relation

PARENT	
CNAME	PNAME
Smith John Jr	Smith John
Smith John Jr	Smith Mary
Rogers Charles	Rogers Linda
Rogers Linda	Jones David
Rogers Linda	Jones Mary
Smith Mary	Ford Albert
Cramer Steven	Cramer William

GRANDPARENT	
CNAME	PNAME
Rogers Charles	Jones David
Rogers Charles	Jones Mary
Smith John Jr	Ford Albert

Relational calculus query to find the grandparent(s) of Charles Rogers

- **GET(X) : GRANDPARENT(Rogers Charles, X)**

REPRESENTATION IN A DEDUCTIVE DATABASE

Extensional database

PARENT	
CNAME	PNAME
Smith John Jr	Smith John
Smith John Jr	Smith Mary
Rogers Charles	Rogers Linda
Rogers Linda	Jones David
Rogers Linda	Jones Mary
Smith Mary	Ford Albert
Cramer Steven	Cramer William

Intensional database

∀X∀Y∀Z PARENT(X,Y)

&

PARENT(Y,Z)

→

GRANDPARENT(X,Z)

Relational calculus query to find the grandparent(s) of Charles Rogers

- **GET(X) : GRANDPARENT(Rogers Charles, X)**

REPRESENTATION AS LOGICAL FORMULAS

PARENT(Smith John Jr, Smith John)

PARENT(Smith John Jr, Smith Mary)

PARENT(Rogers Charles, Rogers Linda)

PARENT(Rogers Linda, Jones David)

PARENT(Rogers Linda, Jones Mary)

PARENT(Smith Mary, Ford Albert)

PARENT(Cramer Steven, Cramer William)

$\forall X \forall Y \forall Z$ PARENT(X,Y) & PARENT(Y,Z) $\rightarrow$ GRANDPARENT(X,Z)

PROOF OF ASSERTIONS WITH THE DATABASE

a) "Is Rogers Charles a parent of Rogers Linda?"

PROOF OF ASSERTIONS WITH THE DATABASE

a) "Is Rogers Charles a parent of Rogers Linda?"
¬PARENT(Rogers Charles, Rogers Linda)

PROOF OF ASSERTIONS WITH THE DATABASE

a) "Is Rogers Charles a parent of Rogers Linda?"

¬PARENT(Rogers Charles, Rogers Linda)

PARENT(Rogers Charles, Rogers Linda)

[]

PROOF OF ASSERTIONS WITH THE DATABASE

a) "Is Rogers Charles a parent of Rogers Linda?"

¬PARENT(Rogers Charles, Rogers Linda)

PARENT(Rogers Charles, Rogers Linda)

[]

b) "Is Rogers Charles a grandparent of Jones Mary?"

PROOF OF ASSERTIONS WITH THE DATABASE

a) **"Is Rogers Charles a parent of Rogers Linda?"**

¬PARENT(Rogers Charles, Rogers Linda)

PARENT(Rogers Charles, Rogers Linda)

[]

b) **"Is Rogers Charles a grandparent of Jones Mary?"**

¬GRANDPARENT(Rogers Charles, Jones Mary)

PROOF OF ASSERTIONS WITH THE DATABASE

a) **“Is Rogers Charles a parent of Rogers Linda?”**

¬PARENT(Rogers Charles, Rogers Linda)

PARENT(Rogers Charles, Rogers Linda)

[]

b) **“Is Rogers Charles a grandparent of Jones Mary?”**

¬GRANDPARENT(Rogers Charles, Jones Mary)

¬PARENT(X,Y) ∨ ¬ PARENT(Y,Z) ∨ GRANDPARENT(X,Z)

PROOF OF ASSERTIONS WITH THE DATABASE

a) **"Is Rogers Charles a parent of Rogers Linda?"**

¬PARENT(Rogers Charles, Rogers Linda)

PARENT(Rogers Charles, Rogers Linda)

[]

b) **"Is Rogers Charles a grandparent of Jones Mary?"**

¬GRANDPARENT(Rogers Charles, Jones Mary)

¬PARENT(X,Y) ∨ ¬ PARENT(Y,Z) ∨ GRANDPARENT(X,Z)

¬PARENT(Rogers Charles,Y) ∨ ¬PARENT(Y, Jones Mary)

PROOF OF ASSERTIONS WITH THE DATABASE

a) "Is Rogers Charles a parent of Rogers Linda?"

¬PARENT(Rogers Charles, Rogers Linda)

PARENT(Rogers Charles, Rogers Linda)

[]

b) "Is Rogers Charles a grandparent of Jones Mary?"

¬GRANDPARENT(Rogers Charles, Jones Mary)

¬PARENT(X,Y) ∨ ¬ PARENT(Y,Z) ∨ GRANDPARENT(X,Z)

¬PARENT(Rogers Charles,Y) ∨ ¬PARENT(Y, Jones Mary)

PARENT(Rogers Charles, Rogers Linda)

PROOF OF ASSERTIONS WITH THE DATABASE

a) **“Is Rogers Charles a parent of Rogers Linda?”**

¬PARENT(Rogers Charles, Rogers Linda)

PARENT(Rogers Charles, Rogers Linda)

[]

b) **“Is Rogers Charles a grandparent of Jones Mary?”**

¬GRANDPARENT(Rogers Charles, Jones Mary)

¬PARENT(X,Y) ∨ ¬ PARENT(Y,Z) ∨ GRANDPARENT(X,Z)

¬PARENT(Rogers Charles,Y) ∨ ¬PARENT(Y, Jones Mary)

PARENT(Rogers Charles, Rogers Linda)

¬PARENT(Rogers Linda, Jones Mary)

PROOF OF ASSERTIONS WITH THE DATABASE

a) “Is Rogers Charles a parent of Rogers Linda?”

¬PARENT(Rogers Charles, Rogers Linda)

PARENT(Rogers Charles, Rogers Linda)

[]

b) “Is Rogers Charles a grandparent of Jones Mary?”

¬GRANDPARENT(Rogers Charles, Jones Mary)

¬PARENT(X,Y) ∨ ¬ PARENT(Y,Z) ∨ GRANDPARENT(X,Z)

¬PARENT(Rogers Charles,Y) ∨ ¬PARENT(Y, Jones Mary)

PARENT(Rogers Charles, Rogers Linda)

¬PARENT(Rogers Linda, Jones Mary)

PARENT(Rogers Linda, Jones Mary)

[]

REPRESENTATION WITH TWO TABLES

Extensional database

PARENT	
CNAME	PNAME
Smith John Jr	Smith John
Smith John Jr	Smith Mary
Rogers Charles	Rogers Linda
Rogers Linda	Jones David
Rogers Linda	Jones Mary
Smith Mary	Ford Albert
Cramer Steven	Cramer William

MALE
NAME
Smith John Jr
Smith John
Rogers Charles
Jones David
Ford Albert
Cramer Steven
Cramer William

Intensional database

∀X∀Y PARENT(X,Y) & MALE(Y) → FATHER(X,Y)

∀X∀Y∀Z PARENT(X,Y) & PARENT(Y,Z) → GRANDPARENT(X,Z)

∀X∀Y∀Z GRANDPARENT(X,Y) & FATHER(Z,Y) → GRANDFATHER(X,Y)

ANSWERING A QUERY WITH THE DATABASE (1)

“Who are the grandparents of Rogers Charles?”

ANSWERING A QUERY WITH THE DATABASE (1)

"Who are the grandparents of Rogers Charles?"

Finding instantiations for the variable in the query representation

¬GRANDPARENT(Rogers Charles, V)

ANSWERING A QUERY WITH THE DATABASE (1)

“Who are the grandparents of Rogers Charles?”

Finding instantiations for the variable in the query representation

¬GRANDPARENT(Rogers Charles, V)

¬PARENT(X,Y) ∨ ¬ PARENT(Y,Z) ∨ GRANDPARENT(X,Z)

ANSWERING A QUERY WITH THE DATABASE (1)

"Who are the grandparents of Rogers Charles?"

Finding instantiations for the variable in the query representation

¬GRANDPARENT(Rogers Charles, V)

¬PARENT(X,Y) ∨ ¬ PARENT(Y,Z) ∨ GRANDPARENT(X,Z)

¬PARENT(Rogers Charles,Y) ∨ ¬PARENT(Y, V)

ANSWERING A QUERY WITH THE DATABASE (1)

“Who are the grandparents of Rogers Charles?”

Finding instantiations for the variable in the query representation

¬GRANDPARENT(Rogers Charles, V)

¬PARENT(X,Y) ∨ ¬ PARENT(Y,Z) ∨ GRANDPARENT(X,Z)

¬PARENT(Rogers Charles,Y) ∨ ¬PARENT(Y, V)

PARENT(Rogers Charles, Rogers Linda)

ANSWERING A QUERY WITH THE DATABASE (1)

"Who are the grandparents of Rogers Charles?"

Finding instantiations for the variable in the query representation

¬GRANDPARENT(Rogers Charles, V)

¬PARENT(X,Y) ∨ ¬ PARENT(Y,Z) ∨ GRANDPARENT(X,Z)

¬PARENT(Rogers Charles,Y) ∨ ¬PARENT(Y, V)

PARENT(Rogers Charles, Rogers Linda)

¬PARENT(Rogers Linda, V)

ANSWERING A QUERY WITH THE DATABASE (1)

“Who are the grandparents of Rogers Charles?”

Finding instantiations for the variable in the query representation

¬GRANDPARENT(Rogers Charles, V)

¬PARENT(X,Y) ∨ ¬ PARENT(Y,Z) ∨ GRANDPARENT(X,Z)

¬PARENT(Rogers Charles,Y) ∨ ¬PARENT(Y, V)

PARENT(Rogers Charles, Rogers Linda)

¬PARENT(Rogers Linda, V)

PARENT(Rogers Linda, Jones Mary)

[]

ANSWERING A QUERY WITH THE DATABASE (1)

"Who are the grandparents of Rogers Charles?"

Finding instantiations for the variable in the query representation

¬GRANDPARENT(Rogers Charles, V)

¬PARENT(X,Y) ∨ ¬ PARENT(Y,Z) ∨ GRANDPARENT(X,Z)

¬PARENT(Rogers Charles,Y) ∨ ¬PARENT(Y, V)

PARENT(Rogers Charles, Rogers Linda)

¬PARENT(Rogers Linda, V)

PARENT(Rogers Linda, Jones Mary)

PARENT(Rogers Linda, Jones David)

[] []

ANSWERING A QUERY WITH THE DATABASE (2)

"Who is the grandfather of Rogers Charles?"

ANSWERING A QUERY WITH THE DATABASE (2)

"Who is the grandfather of Rogers Charles?"

¬GRANDFATHER(Rogers Charles, X)

ANSWERING A QUERY WITH THE DATABASE (2)

"Who is the grandfather of Rogers Charles?"

¬GRANDFATHER(Rogers Charles, X)

¬GRANDPARENT(V,Y) ∨ ¬ FATHER(Z,Y) ∨ GRANDFATHER(V, Y)

ANSWERING A QUERY WITH THE DATABASE (2)

"Who is the grandfather of Rogers Charles?"

¬GRANDFATHER(Rogers Charles, X)

\ / ¬GRANDPARENT(V,Y) ∨ ¬ FATHER(Z,Y) ∨ GRANDFATHER(V, Y)

¬GRANDPARENT(Rogers Charles,X) ∨ ¬FATHER(Z, X)

ANSWERING A QUERY WITH THE DATABASE (2)

"Who is the grandfather of Rogers Charles?"

¬GRANDFATHER(Rogers Charles, X)

¬GRANDPARENT(V,Y) ∨ ¬ FATHER(Z,Y) ∨ GRANDFATHER(V, Y)

¬GRANDPARENT(Rogers Charles,X) ∨ ¬FATHER(Z, X)

¬PARENT(V,Y) ∨ ¬ PARENT(Y,W) ∨ GRANDPARENT(V, W)

ANSWERING A QUERY WITH THE DATABASE (2)

"Who is the grandfather of Rogers Charles?"

¬GRANDFATHER(Rogers Charles, X)

¬GRANDPARENT(V,Y) ∨ ¬ FATHER(Z,Y) ∨ GRANDFATHER(V, Y)

¬GRANDPARENT(Rogers Charles,X) ∨ ¬FATHER(Z, X)

¬PARENT(V,Y) ∨ ¬ PARENT(Y,W) ∨ GRANDPARENT(V, W)

¬PARENT(Rogers Charles,Y) ∨ ¬PARENT(Y, X) ∨ ¬FATHER(Z, X)

ANSWERING A QUERY WITH THE DATABASE (2)

“Who is the grandfather of Rogers Charles?”

¬GRANDFATHER(Rogers Charles, X)

¬GRANDPARENT(V,Y) ∨ ¬ FATHER(Z,Y) ∨ GRANDFATHER(V, Y)

¬GRANDPARENT(Rogers Charles,X) ∨ ¬FATHER(Z, X)

¬PARENT(V,Y) ∨ ¬ PARENT(Y,W) ∨ GRANDPARENT(V, W)

¬PARENT(Rogers Charles,Y) ∨ ¬PARENT(Y, X) ∨ ¬FATHER(Z, X)

¬PARENT(V,W) ∨ ¬ MALE(W) ∨ FATHER(V, W)

ANSWERING A QUERY WITH THE DATABASE (2)

"Who is the grandfather of Rogers Charles?"

¬GRANDFATHER(Rogers Charles, X)

¬GRANDPARENT(V,Y) ∨ ¬ FATHER(Z,Y) ∨ GRANDFATHER(V, Y)

¬GRANDPARENT(Rogers Charles,X) ∨ ¬FATHER(Z, X)

¬PARENT(V,Y) ∨ ¬ PARENT(Y,W) ∨ GRANDPARENT(V, W)

¬PARENT(Rogers Charles,Y) ∨ ¬PARENT(Y, X) ∨ ¬FATHER(Z, X)

¬PARENT(V,W) ∨ ¬ MALE(W) ∨ FATHER(V, W)

¬PARENT(Rogers Charles,Y) ∨ ¬PARENT(Y, X) ∨ ¬PARENT(Z, X) ∨ ¬MALE(X)

ANSWERING A QUERY WITH THE DATABASE (2)

“Who is the grandfather of Rogers Charles?”

¬GRANDFATHER(Rogers Charles, X)

¬GRANDPARENT(V,Y) ∨ ¬ FATHER(Z,Y) ∨ GRANDFATHER(V, Y)

¬GRANDPARENT(Rogers Charles,X) ∨ ¬FATHER(Z, X)

¬PARENT(V,Y) ∨ ¬ PARENT(Y,W) ∨ GRANDPARENT(V, W)

¬PARENT(Rogers Charles,Y) ∨ ¬PARENT(Y, X) ∨ ¬FATHER(Z, X)

¬PARENT(V,W) ∨ ¬ MALE(W) ∨ FATHER(V, W)

¬PARENT(Rogers Charles,Y) ∨ ¬PARENT(Y, X) ∨ ¬PARENT(Z, X) ∨ ¬MALE(X)

PARENT(Rogers Charles, Rogers Linda)

ANSWERING A QUERY WITH THE DATABASE (2)

“Who is the grandfather of Rogers Charles?”

¬GRANDFATHER(Rogers Charles, X)

¬GRANDPARENT(V,Y) ∨ ¬ FATHER(Z,Y) ∨ GRANDFATHER(V, Y)

¬GRANDPARENT(Rogers Charles,X) ∨ ¬FATHER(Z, X)

¬PARENT(V,Y) ∨ ¬ PARENT(Y,W) ∨ GRANDPARENT(V, W)

¬PARENT(Rogers Charles,Y) ∨ ¬PARENT(Y, X) ∨ ¬FATHER(Z, X)

¬PARENT(V,W) ∨ ¬ MALE(W) ∨ FATHER(V, W)

¬PARENT(Rogers Charles,Y) ∨ ¬PARENT(Y, X) ∨ ¬PARENT(Z, X) ∨ ¬MALE(X)

PARENT(Rogers Charles, Rogers Linda)

¬PARENT(Rogers Linda, X) ∨ ¬PARENT(Z, X) ∨ ¬MALE(X)

ANSWERING A QUERY WITH THE DATABASE (2)

"Who is the grandfather of Rogers Charles?"

¬GRANDFATHER(Rogers Charles, X)

¬GRANDPARENT(V,Y) ∨ ¬ FATHER(Z,Y) ∨ GRANDFATHER(V, Y)

¬GRANDPARENT(Rogers Charles,X) ∨ ¬FATHER(Z, X)

¬PARENT(V,Y) ∨ ¬ PARENT(Y,W) ∨ GRANDPARENT(V, W)

¬PARENT(Rogers Charles,Y) ∨ ¬PARENT(Y, X) ∨ ¬FATHER(Z, X)

¬PARENT(V,W) ∨ ¬ MALE(W) ∨ FATHER(V, W)

¬PARENT(Rogers Charles,Y) ∨ ¬PARENT(Y, X) ∨ ¬PARENT(Z, X) ∨ ¬MALE(X)

PARENT(Rogers Charles, Rogers Linda)

¬PARENT(Rogers Linda, X) ∨ ¬PARENT(Z, X) ∨ ¬MALE(X)

¬PARENT(Rogers Linda, X) ∨ ¬MALE(X)

ANSWERING A QUERY WITH THE DATABASE (2)

"Who is the grandfather of Rogers Charles?"

¬GRANDFATHER(Rogers Charles, X)

¬GRANDPARENT(V,Y) ∨ ¬ FATHER(Z,Y) ∨ GRANDFATHER(V, Y)

¬GRANDPARENT(Rogers Charles,X) ∨ ¬FATHER(Z, X)

¬PARENT(V,Y) ∨ ¬ PARENT(Y,W) ∨ GRANDPARENT(V, W)

¬PARENT(Rogers Charles,Y) ∨ ¬PARENT(Y, X) ∨ ¬FATHER(Z, X)

¬PARENT(V,W) ∨ ¬ MALE(W) ∨ FATHER(V, W)

¬PARENT(Rogers Charles,Y) ∨ ¬PARENT(Y, X) ∨ ¬PARENT(Z, X) ∨ ¬MALE(X)

PARENT(Rogers Charles, Rogers Linda)

¬PARENT(Rogers Linda, X) ∨ ¬PARENT(Z, X) ∨ ¬MALE(X)

¬PARENT(Rogers Linda, X) ∨ ¬MALE(X)

PARENT(Rogers Linda, Jones David)

ANSWERING A QUERY WITH THE DATABASE (2)

"Who is the grandfather of Rogers Charles?"

¬GRANDFATHER(Rogers Charles, X)

¬GRANDPARENT(V,Y) ∨ ¬ FATHER(Z,Y) ∨ GRANDFATHER(V, Y)

¬GRANDPARENT(Rogers Charles,X) ∨ ¬FATHER(Z, X)

¬PARENT(V,Y) ∨ ¬ PARENT(Y,W) ∨ GRANDPARENT(V, W)

¬PARENT(Rogers Charles,Y) ∨ ¬PARENT(Y, X) ∨ ¬FATHER(Z, X)

¬PARENT(V,W) ∨ ¬ MALE(W) ∨ FATHER(V, W)

¬PARENT(Rogers Charles,Y) ∨ ¬PARENT(Y, X) ∨ ¬PARENT(Z, X) ∨ ¬MALE(X)

PARENT(Rogers Charles, Rogers Linda)

¬PARENT(Rogers Linda, X) ∨ ¬PARENT(Z, X) ∨ ¬MALE(X)

¬PARENT(Rogers Linda, X) ∨ ¬MALE(X)

PARENT(Rogers Linda, Jones David)

¬MALE(Jones David)

ANSWERING A QUERY WITH THE DATABASE (2)

"Who is the grandfather of Rogers Charles?"

¬GRANDFATHER(Rogers Charles, X)

¬GRANDPARENT(V,Y) ∨ ¬ FATHER(Z,Y) ∨ GRANDFATHER(V, Y)

¬GRANDPARENT(Rogers Charles,X) ∨ ¬FATHER(Z, X)

¬PARENT(V,Y) ∨ ¬ PARENT(Y,W) ∨ GRANDPARENT(V, W)

¬PARENT(Rogers Charles,Y) ∨ ¬PARENT(Y, X) ∨ ¬FATHER(Z, X)

¬PARENT(V,W) ∨ ¬ MALE(W) ∨ FATHER(V, W)

¬PARENT(Rogers Charles,Y) ∨ ¬PARENT(Y, X) ∨ ¬PARENT(Z, X) ∨ ¬MALE(X)

PARENT(Rogers Charles, Rogers Linda)

¬PARENT(Rogers Linda, X) ∨ ¬PARENT(Z, X) ∨ ¬MALE(X)

¬PARENT(Rogers Linda, X) ∨ ¬MALE(X)

PARENT(Rogers Linda, Jones David)

¬MALE(Jones David)

MALE(Jones David)

[]

THE STATE OF DEDUCTIVE DATABASES

Problems with deductive databases

- **No commercial deductive database (one company producing DDBMS had to close)**
- **Prolog implementations are much faster than deductive databases**

Use of ideas of deductive databases

- **Prolog is used successfully in industry**
- **Constraint logic programming is very successful in industry**
- **Ideas of deductive databases used in extensions to standard relational databases**
- **Answer set programming as a new logic programming formalism**

A

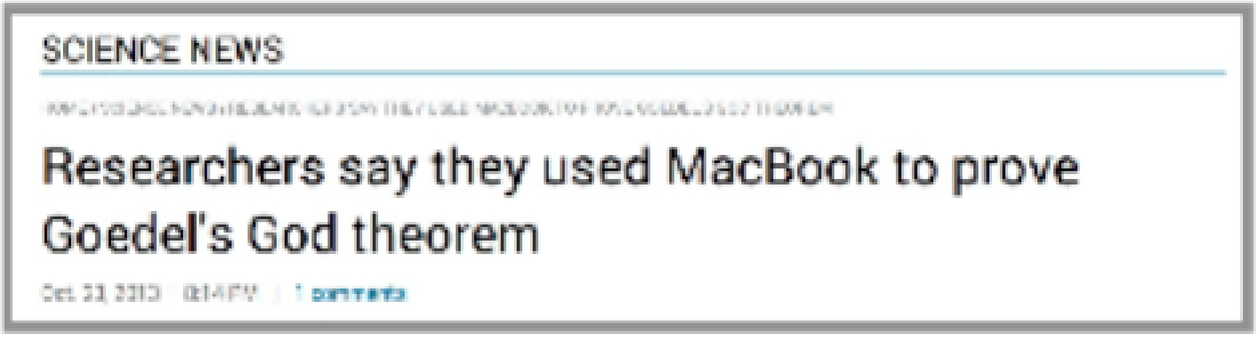
Proof

of

God

(Christoph Benzmüller and Bruno Woltzenlogel Paleo, 2013)

IN A NUTSHELL


SCIENCE NEWS

Researchers say they used MacBook to prove Goedel's God theorem

Techniques

- **Advanced proof techniques required (higher order, modal logic)**
- **Progress in use of logics obtained**

Media interest

- **Germany: Spiegel Online, FAZ, Die Welt, …**
- **International: Austria, Italy, India, US, …**

HISTORICAL BACKGROUND

Ontological argument

- **Deductive argument (for the existence of God)**
- **Starting from premises, justified by pure reasoning**

Rich history of ontological arguments

- **Pro: Descartes, Leibniz, Hegel, Gödel, …**
- **Against: Th. Aquinas, Kant, Frege, …**

Gödels notion of god

"A God-like being possesses all positive properties"

->

"(Necessarily) God exists"

proved by Gödel on two hand-written pages

SCOTT'S VERSION OF GÖDEL'S AXIOMS, DEFINITIONS AND THEOREMS

A1 Either a property or its negation is positive, but not both: $\forall\phi[P(\neg\phi) \leftrightarrow \neg P(\phi)]$

A2 A property necessarily implied by a positive property is positive: $\forall\phi\forall\psi[(P(\phi) \wedge \Box\forall x[\phi(x) \to \psi(x)]) \to P(\psi)]$

T1 Positive properties are possibly exemplified: $\forall\varphi[P(\varphi) \to \Diamond\exists x\varphi(x)]$

D1 A *God-like* being possesses all positive properties: $G(x) \leftrightarrow \forall\phi[P(\phi) \to \phi(x)]$

A3 The property of being God-like is positive: $P(G)$

C Possibly, God exists: $\Diamond\exists xG(x)$

A4 Positive properties are necessarily positive: $\forall\phi[P(\phi) \to \Box\, P(\phi)]$

D2 An *essence* of an individual is a property possessed by it and necessarily implying any of its properties: $\phi\ ess.\ x \leftrightarrow \phi(x) \wedge \forall\psi(\psi(x) \to \Box\forall y(\phi(y) \to \psi(y)))$

T2 Being God-like is an essence of any God-like being: $\forall x[G(x) \to G\ ess.\ x]$

D3 *Necessary existence* of an individual is the necessary exemplification of all its essences: $NE(x) \leftrightarrow \forall\phi[\phi\ ess.\ x \to \Box\exists y\phi(y)]$

A5 Necessary existence is a positive property: $P(NE)$

T3 Necessarily, God exists: $\Box\exists xG(x)$

PROOF OVERVIEW (IN NATURAL DEDUCTION STYLE)

A2: $\forall\varphi.\forall\psi.[(P(\varphi) \land \Box\forall x.[\varphi(x) \rightarrow \psi(x)]) \rightarrow P(\psi)]$ **A1a**: $\forall\varphi.[P(\neg\varphi) \rightarrow \neg P(\varphi)]$

A3: $P(G)$ **T1**: $\forall\varphi.[P(\varphi) \rightarrow \Diamond\exists x.\varphi(x)]$

C1: $\Diamond\exists z.G(z)$

A1b: $\forall\varphi.[\neg P(\varphi) \rightarrow P(\neg\varphi)]$ **A4**: $\forall\varphi.[P(\varphi) \rightarrow \Box P(\varphi)]$

T2: $\forall y.[G(y) \rightarrow G \text{ } ess \text{ } y]$ **A5**: $P(NE)$

L1: $\exists z.G(z) \rightarrow \Box\exists x.G(x)$

$\Diamond\exists z.G(z) \rightarrow \Diamond\Box\exists x.G(x)$ **S5**: $\forall\xi.[\Diamond\Box\xi \rightarrow \Box\xi]$

L2: $\Diamond\exists z.G(z) \rightarrow \Box\exists x.G(x)$

C1: $\Diamond\exists z.G(z)$ **L2**: $\Diamond\exists z.G(z) \rightarrow \Box\exists x.G(x)$

T3: $\Box\exists x.G(x)$

PROOF DESIGN

State-of-the-art

- **No prover for higher order modal logic exists**
- **Several (increasingly better and coordinated) provers for higher order logic exist (interactive and automated ones)**

Overall strategy

- **Embedding in higher order classical logic (based on experience with embedding first-order modal logic in higher order logic)**
- **Making use of higher-order logic theorem provers**
- **Interactive proof oriented on human-designed natural deduction proof**

Assessment

A fully automated proof may be possible in about 3 years

Benzmüller

EMBEDDING IN HIGHER-ORDER LOGIC

QML $\varphi, \psi \; ::= \; \ldots \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \rightarrow \psi \mid \Box\varphi \mid \Diamond\varphi \mid \forall x\,\varphi \mid \exists x\,\varphi \mid \forall P\,\varphi$

HOL $s, t \; ::= \; C \mid x \mid \lambda x s \mid s\,t \mid \neg s \mid s \vee t \mid \forall x\,t$

QML in HOL: QML formulas φ are mapped to HOL predicates $\varphi_{\iota\rightarrow o}$

$\neg$	=	$\lambda\varphi_{\iota\rightarrow o}\lambda s_\iota \neg\varphi s$
$\wedge$	=	$\lambda\varphi_{\iota\rightarrow o}\lambda\psi_{\iota\rightarrow o}\lambda s_\iota(\varphi s \wedge \psi s)$
$\rightarrow$	=	$\lambda\varphi_{\iota\rightarrow o}\lambda\psi_{\iota\rightarrow o}\lambda s_\iota(\neg\varphi s \vee \psi s)$
$\Box$	=	$\lambda\varphi_{\iota\rightarrow o}\lambda s_\iota \forall u_\iota\,(\neg rsu \vee \varphi u)$
$\Diamond$	=	$\lambda\varphi_{\iota\rightarrow o}\lambda s_\iota \exists u_\iota\,(rsu \wedge \varphi u)$
$\forall$	=	$\lambda h_{\mu\rightarrow(\iota\rightarrow o)}\lambda s_\iota \forall d_\mu\, hds$
$\exists$	=	$\lambda h_{\mu\rightarrow(\iota\rightarrow o)}\lambda s_\iota \exists d_\mu\, hds$
$\forall$	=	$\lambda H_{(\mu\rightarrow(\iota\rightarrow o))\rightarrow(\iota\rightarrow o)}\lambda s_\iota \forall d_\mu\, Hds$
valid	=	$\lambda\varphi_{\iota\rightarrow o}\forall w_\iota \varphi w$

Ax

The equations in Ax are given as axioms to the HOL provers!

(Remark: Note that we are here dealing with constant domain quantification)

COURSE OF THE PROOF

Subproof	*Prover responsible*
Checking consistency	**Nitpick (model checker)**
Checking consistency Gödels original definition of D2	**LEO II (ATP)**
Proving T1 (positive properties ev. exemplified) is a theorem	**LEO II (ATP)**
Proving C (possibly, God exists) is a theorem	**LEO II (ATP)**
Proving T2 (being God-like is an essence …) is a theorem	**LEO II (ATP)**
Proving T3 (necessarily God exists) is a theorem	**LEO II (ATP)**
Proving C2 (necessarily God exists) is a theorem	**LEO II (ATP)**
Checking axioms are consistent	**Nitpick (model checker)**
Checking Gödels original axioms are inconsistent	**LEO II (ATP)**
Checking modal collaps	**LEO II (ATP)**
Checking "flawlessness of God"	**LEO II (ATP)**
Proving Monotheism	**TPS (ATP)**

CRITICISM & OUTLOOK

Problematic assumptions

- **Everything that is the case is so necessarily. $\forall P.[P \rightarrow \Box P]$**
  (follows from T2, T3, D2, proved by higher order ATPs)
  Then everything is determined, there is no free will …
- **Either a property or its negation is positive**
  in the morale sense, according to Gödel

Results

- **Powerful infrastucture to reason in higher-order modal logic**
- **Several insights about the strength of logics needed or not needed**
- **Difficult benchmark problems for higher-order theorem provers**
- **Major step towards computer-assisted theoretical philosophy**
- **Further ontological arguments to be tested (in particular, related to Gödel)**

(see http://page.mi.fu-berlin.de/cbenzmueller/, link presentations)

STATE OF AFFAIRS OF THEOREM PROVERS

Capabilities

- **Occasional success with proofs of prominent theorems**
  (usually tedious and extremely longish, but first known formal result)
- **Some specialized provers**
  (taxonomic reasoners, equation provers)
- **Considerable progress in efficiency recently**

Variety of uses

- **Remote access to several ATPs (first-order, higher order)**
- **Calling several (distinct) provers in parallel (hoping at least one succeeds)**
- **Combining reasoning techniques (proving + computer algebra)**
- **Interactive proving (adding control for the prover, software verification)**
- **Proof planning – provers supported by proof schemas that encapsulate knowledge**

ANSWERING INTENSIONAL QUESTIONS (1)

Alternative query formulations and answers

- ***Which*** **managers at IBM earn more than 1,000,000 $?**

  Smith, Miller, and Jones.

- ***What*** **managers at IBM earn more than 1,000,000 $?**

  Managers of rank A.

Building intensional answers

- **Based on *deduction***

  Exploiting rule traces

- **Based on *induction (inductive logic programming)***

  Reasoning over commonalities of extensional sets

ANSWERING INTENSIONAL QUESTIONS (2)

Motivation

- **Checking consistency constraints (debugging a database)**
- **Producing informative descriptions rather than pure enumerations**

Techniques for building intensional answers – idea

- **Building a description of one item**
- **Successively adapting this description, such that**
  - **increasingly more items in the "solution set" are covered**
  - **no element outside the "solution set" is covered**
  - **performing simplifications/generalizations whenever appropriate**

This task is the same as building referring expressions identifying sets of objects!

ANSWERING INTENSIONAL QUESTIONS – EXAMPLES

Checking consistency

"Which states have a capital?" – "All states."

"Which cities have inhabitants?" – "All German cities."
(Swiss cities on the border not modeled)

Producing informative descriptions

"Which cities are in a state that borders Austria?" – "All cities located in Bayern."

"Which cities are bigger than München?" (Berlin, Hamburg)

"All cities located at highway A24 and which a river flows through."

Limitations

Commonalities must be "total" (no disjunctions, no exceptions)

No concept indicating what is "of interest"

PARSING AS DEDUCTION

Connection between parsing and deduction

- **Axiomatization of context-free grammars in definite clauses (subset of first-order logic)**
- **Identification of context-free parsing with deduction for a restricted class of definite clauses**
- **Extension to larger classes of definite clauses by replacing atomic grammar symbols by complex ones matched by unification – constraints specified by an argument**
- **Further extended to unification grammars**

Parsing algorithms

- ***Offline* – constraints *after* context-free parsing**
- ***Online* – constraints *during* context-free parsing (considered here)**

DEFINITE CLAUSES

Definitions – definite clauses

$P \Leftarrow Q_1 \;\&\; \dots \;\&\; Q_n$

- **P is true if Q_1 and … and Q_n are true**
- **P is *positive* literal or *head***
- **$Q_1 \dots Q_n$ are *negative* literals or *body***
- **Literals have the form $p(t_1,\dots,t_k)$ with predicate p (arity k) and t_i as arguments (terms)**

Definitions – a program

- **A set of clauses (input clauses) is a *program***
- **A program defines the relations between the predicates appearing in the heads of the clauses**
- **A goal statement $\Leftarrow P$ requires finding provable instances of P**

DEFINITE CLAUSES

Definite clause grammars

Context-free rule $X \rightarrow \alpha_1 \ldots \alpha_n$

translated to definite clause $X(S_0,S_n) \Leftarrow \alpha_1(S_0,S_1) \; \& \ldots \& \; \alpha_n(S_{n-1},S_n)$

with variables S_i being string arguments (positions in the input string)

- **Generalization by adding predicate arguments to string arguments**

Deduction in definite clauses

Resolution

(1) $B \Leftarrow A_1 \; \& \ldots \& \; A_m$

(2) $C \Leftarrow D_1 \; \& \ldots \& \; D_i \; \& \ldots \& \; D_n$

if B and D_i are unifiable by substitution σ, infer

(3) $\sigma[C \Leftarrow D_1 \; \& \ldots \& \; D_{i-1} \; \& \; A_1 \; \& \ldots \& \; A_m \ldots \& \; D_{i+1} \ldots \& \; D_n]$

Clause (3) is a derived clause, resolvent from (1) and (2)

EARLEY DEDUCTION

Definitions

- **Definite clauses divided into *program* and *state***
- **Program – set of *input* clauses, fixed**
- **State – set of *derived* clauses, nonunit clauses with one negative literal selected (initially the goal)**

Inference rules

- ***Instantiation* – selected literal of some clause unifies with a positive literal of a *nonunit* clause C in the program, deriving the instantiation $\sigma[C]$**

  (σ is the most general unifier of the two literals)
- ***Reduction* – selected literal of some clause unifies with a *unit* clause in the program or in the current state, deriving the clause $\sigma[C']$**

  C' is C minus the selected literal (σ is the most general unifier of the two literals)

Techniques

- **Mixed top-down bottom-up mechanism**
- **Blockage of derived clauses subsumed by the state**
- **Handling gaps, dependencies by extra arguments**

EXAMPLE DEDUCTION PROOF

Context free grammar			***Definite clause program***			
S	**→**	**NP VP**	**s(S0,S)**	**⇐**	**np(S0,S1) & vp(S1,S)**	**(20)**
NP	**→**	**Det N**	**np(S0,S)**	**⇐**	**det(S0,S1) & n(S1,S)**	**(21)**
Det	**→**	**NP Gen**	**det(S0,S)**	**⇐**	**np(S0,S1) & gen(S1,S)**	**(22)**
Det	**→**	**Art**	**det(S0,S)**	**⇐**	**art(S0,S)**	**(23)**
Det	**→**	**A**	**det(S0,S)**			**(24)**
VP	**→**	**V NP**	**vp(S0,S)**	**⇐**	**v(S0,S1) & np(S1,S)**	**(25)**

Lexical categories
of the sentence

$_0$**Agatha**$_1$**'s**$_2$**husband**$_3$**hit**$_4$**Ulrich**$_5$

represented by the unit clauses

n(0,1) gen(1,2) n(2,3) v(3,4) n(4,5) **(26)**

goal statement ans ⇐ s(0,5) provable, if (26) is a sentence

EXAMPLE DEDUCTION PROOF (1)

s(S0,S)	**⇐**	**np(S0,S1) & vp(S1,S)**		**(20)**
np(S0,S)	**⇐**	**det(S0,S1) & n(S1,S)**		**(21)**
det(S0,S)	**⇐**	**np(S0,S1) & gen(S1,S)**		**(22)**
det(S0,S)	**⇐**	**art(S0,S)**		**(23)**
det(S0,S)				**(24)**
vp(S0,S)	**⇐**	**v(S0,S1) & np(S1,S)**		**(25)**
ans	**⇐**	**s(0,5)**	**goal statement**	**(33)**

EXAMPLE DEDUCTION PROOF (1)

s(S0,S)	**⇐**	**np(S0,S1) & vp(S1,S)**		**(20)**
np(S0,S)	**⇐**	**det(S0,S1) & n(S1,S)**		**(21)**
det(S0,S)	**⇐**	**np(S0,S1) & gen(S1,S)**		**(22)**
det(S0,S)	**⇐**	**art(S0,S)**		**(23)**
det(S0,S)				**(24)**
vp(S0,S)	**⇐**	**v(S0,S1) & np(S1,S)**		**(25)**
ans	**⇐**	**s(0,5)**	**goal statement**	**(33)**
s(0,5)	**⇐**	**np(0,S1) & vp(S1,5)**	**(33) instantiates (20)**	**(34)**

EXAMPLE DEDUCTION PROOF (1)

s(S0,S)	**⇐**	**np(S0,S1) & vp(S1,S)**		**(20)**
np(S0,S)	**⇐**	**det(S0,S1) & n(S1,S)**		**(21)**
det(S0,S)	**⇐**	**np(S0,S1) & gen(S1,S)**		**(22)**
det(S0,S)	**⇐**	**art(S0,S)**		**(23)**
det(S0,S)				**(24)**
vp(S0,S)	**⇐**	**v(S0,S1) & np(S1,S)**		**(25)**
ans	**⇐**	**s(0,5)**	**goal statement**	**(33)**
s(0,5)	**⇐**	**np(0,S1) & vp(S1,5)**	**(33) instantiates (20)**	**(34)**
np(0,S)	**⇐**	**det(0,S1) & n(S1,S)**	**(34) instantiates (21)**	**(35)**

EXAMPLE DEDUCTION PROOF (1)

s(S0,S)	⇐	np(S0,S1) & vp(S1,S)		(20)
np(S0,S)	⇐	det(S0,S1) & n(S1,S)		(21)
det(S0,S)	⇐	np(S0,S1) & gen(S1,S)		(22)
det(S0,S)	⇐	art(S0,S)		(23)
det(S0,S)				(24)
vp(S0,S)	⇐	v(S0,S1) & np(S1,S)		(25)
ans	⇐	s(0,5)	goal statement	(33)
s(0,5)	⇐	np(0,S1) & vp(S1,5)	(33) instantiates (20)	(34)
np(0,S)	⇐	det(0,S1) & n(S1,S)	(34) instantiates (21)	(35)
det(0,S)	⇐	np(0,S1) & gen(S1,S)	(35) instantiates (22)	(36)

EXAMPLE DEDUCTION PROOF (1)

s(S0,S)	**⇐**	**np(S0,S1) & vp(S1,S)**		**(20)**
np(S0,S)	**⇐**	**det(S0,S1) & n(S1,S)**		**(21)**
det(S0,S)	**⇐**	**np(S0,S1) & gen(S1,S)**		**(22)**
det(S0,S)	**⇐**	**art(S0,S)**		**(23)**
det(S0,S)				**(24)**
vp(S0,S)	**⇐**	**v(S0,S1) & np(S1,S)**		**(25)**
ans	**⇐**	**s(0,5)**	**goal statement**	**(33)**
s(0,5)	**⇐**	**np(0,S1) & vp(S1,5)**	**(33) instantiates (20)**	**(34)**
np(0,S)	**⇐**	**det(0,S1) & n(S1,S)**	**(34) instantiates (21)**	**(35)**
det(0,S)	**⇐**	**np(0,S1) & gen(S1,S)**	**(35) instantiates (22)**	**(36)**
det(0,S)	**⇐**	**art(0,S)**	**(35) instantiates (23)**	**(37)**

EXAMPLE DEDUCTION PROOF (1)

s(S0,S)	⇐	np(S0,S1) & vp(S1,S)		(20)
np(S0,S)	⇐	det(S0,S1) & n(S1,S)		(21)
det(S0,S)	⇐	np(S0,S1) & gen(S1,S)		(22)
det(S0,S)	⇐	art(S0,S)		(23)
det(S0,S)				(24)
vp(S0,S)	⇐	v(S0,S1) & np(S1,S)		(25)
ans	⇐	s(0,5)	goal statement	(33)
s(0,5)	⇐	np(0,S1) & vp(S1,5)	(33) instantiates (20)	(34)
np(0,S)	⇐	det(0,S1) & n(S1,S)	(34) instantiates (21)	(35)
det(0,S)	⇐	np(0,S1) & gen(S1,S)	(35) instantiates (22)	(36)
det(0,S)	⇐	art(0,S)	(35) instantiates (23)	(37)
np(0,S)	⇐	n(0,S)	(24) reduces (35)	(38)

EXAMPLE DEDUCTION PROOF (1)

s(S0,S)	⇐	np(S0,S1) & vp(S1,S)		(20)
np(S0,S)	⇐	det(S0,S1) & n(S1,S)		(21)
det(S0,S)	⇐	np(S0,S1) & gen(S1,S)		(22)
det(S0,S)	⇐	art(S0,S)		(23)
det(S0,S)				(24)
vp(S0,S)	⇐	v(S0,S1) & np(S1,S)		(25)
ans	⇐	s(0,5)	goal statement	(33)
s(0,5)	⇐	np(0,S1) & vp(S1,5)	(33) instantiates (20)	(34)
np(0,S)	⇐	det(0,S1) & n(S1,S)	(34) instantiates (21)	(35)
det(0,S)	⇐	np(0,S1) & gen(S1,S)	(35) instantiates (22)	(36)
det(0,S)	⇐	art(0,S)	(35) instantiates (23)	(37)
np(0,S)	⇐	n(0,S)	(24) reduces (35)	(38)
np(0,1)			(27) reduces (38)	(39)

EXAMPLE DEDUCTION PROOF (1)

s(S0,S)	⇐	np(S0,S1) & vp(S1,S)		(20)
np(S0,S)	⇐	det(S0,S1) & n(S1,S)		(21)
det(S0,S)	⇐	np(S0,S1) & gen(S1,S)		(22)
det(S0,S)	⇐	art(S0,S)		(23)
det(S0,S)				(24)
vp(S0,S)	⇐	v(S0,S1) & np(S1,S)		(25)
ans	⇐	s(0,5)	goal statement	(33)
s(0,5)	⇐	np(0,S1) & vp(S1,5)	(33) instantiates (20)	(34)
np(0,S)	⇐	det(0,S1) & n(S1,S)	(34) instantiates (21)	(35)
det(0,S)	⇐	np(0,S1) & gen(S1,S)	(35) instantiates (22)	(36)
det(0,S)	⇐	art(0,S)	(35) instantiates (23)	(37)
np(0,S)	⇐	n(0,S)	(24) reduces (35)	(38)
np(0,1)			(27) reduces (38)	(39)
s(0,5)	⇐	vp(1,5)	(39) reduces (34)	(40)

EXAMPLE DEDUCTION PROOF (1)

s(S0,S)	⇐	np(S0,S1) & vp(S1,S)		(20)
np(S0,S)	⇐	det(S0,S1) & n(S1,S)		(21)
det(S0,S)	⇐	np(S0,S1) & gen(S1,S)		(22)
det(S0,S)	⇐	art(S0,S)		(23)
det(S0,S)				(24)
vp(S0,S)	⇐	v(S0,S1) & np(S1,S)		(25)
ans	⇐	s(0,5)	goal statement	(33)
s(0,5)	⇐	np(0,S1) & vp(S1,5)	(33) instantiates (20)	(34)
np(0,S)	⇐	det(0,S1) & n(S1,S)	(34) instantiates (21)	(35)
det(0,S)	⇐	np(0,S1) & gen(S1,S)	(35) instantiates (22)	(36)
det(0,S)	⇐	art(0,S)	(35) instantiates (23)	(37)
np(0,S)	⇐	n(0,S)	(24) reduces (35)	(38)
np(0,1)			(27) reduces (38)	(39)
s(0,5)	⇐	vp(1,5)	(39) reduces (34)	(40)
vp(1,5)	⇐	v(1,S1) & np(S1,5)	(40) instantiates (25)	(41)

EXAMPLE DEDUCTION PROOF (1)

s(S0,S)	⇐	np(S0,S1) & vp(S1,S)		(20)
np(S0,S)	⇐	det(S0,S1) & n(S1,S)		(21)
det(S0,S)	⇐	np(S0,S1) & gen(S1,S)		(22)
det(S0,S)	⇐	art(S0,S)		(23)
det(S0,S)				(24)
vp(S0,S)	⇐	v(S0,S1) & np(S1,S)		(25)
ans	⇐	s(0,5)	goal statement	(33)
s(0,5)	⇐	np(0,S1) & vp(S1,5)	(33) instantiates (20)	(34)
np(0,S)	⇐	det(0,S1) & n(S1,S)	(34) instantiates (21)	(35)
det(0,S)	⇐	np(0,S1) & gen(S1,S)	(35) instantiates (22)	(36)
det(0,S)	⇐	art(0,S)	(35) instantiates (23)	(37)
np(0,S)	⇐	n(0,S)	(24) reduces (35)	(38)
np(0,1)			(27) reduces (38)	(39)
s(0,5)	⇐	vp(1,5)	(39) reduces (34)	(40)
vp(1,5)	⇐	v(1,S1) & np(S1,5)	(40) instantiates (25)	(41)
det(0,S)	⇐	gen(1,5)	(39) reduces (36)	(42)

EXAMPLE DEDUCTION PROOF (1)

s(S0,S)	**⇐**	**np(S0,S1) & vp(S1,S)**		**(20)**
np(S0,S)	**⇐**	**det(S0,S1) & n(S1,S)**		**(21)**
det(S0,S)	**⇐**	**np(S0,S1) & gen(S1,S)**		**(22)**
det(S0,S)	**⇐**	**art(S0,S)**		**(23)**
det(S0,S)				**(24)**
vp(S0,S)	**⇐**	**v(S0,S1) & np(S1,S)**		**(25)**
ans	**⇐**	**s(0,5)**	**goal statement**	**(33)**
s(0,5)	**⇐**	**np(0,S1) & vp(S1,5)**	**(33) instantiates (20)**	**(34)**
np(0,S)	**⇐**	**det(0,S1) & n(S1,S)**	**(34) instantiates (21)**	**(35)**
det(0,S)	**⇐**	**np(0,S1) & gen(S1,S)**	**(35) instantiates (22)**	**(36)**
det(0,S)	**⇐**	**art(0,S)**	**(35) instantiates (23)**	**(37)**
np(0,S)	**⇐**	**n(0,S)**	**(24) reduces (35)**	**(38)**
np(0,1)			**(27) reduces (38)**	**(39)**
s(0,5)	**⇐**	**vp(1,5)**	**(39) reduces (34)**	**(40)**
vp(1,5)	**⇐**	**v(1,S1) & np(S1,5)**	**(40) instantiates (25)**	**(41)**
det(0,S)	**⇐**	**gen(1,5)**	**(39) reduces (36)**	**(42)**
det(0,2)			**(28) reduces (42)**	**(43)**

EXAMPLE DEDUCTION PROOF (1)

s(S0,S)	⇐	np(S0,S1) & vp(S1,S)		(20)
np(S0,S)	⇐	det(S0,S1) & n(S1,S)		(21)
det(S0,S)	⇐	np(S0,S1) & gen(S1,S)		(22)
det(S0,S)	⇐	art(S0,S)		(23)
det(S0,S)				(24)
vp(S0,S)	⇐	v(S0,S1) & np(S1,S)		(25)
ans	⇐	s(0,5)	goal statement	(33)
s(0,5)	⇐	np(0,S1) & vp(S1,5)	(33) instantiates (20)	(34)
np(0,S)	⇐	det(0,S1) & n(S1,S)	(34) instantiates (21)	(35)
det(0,S)	⇐	np(0,S1) & gen(S1,S)	(35) instantiates (22)	(36)
det(0,S)	⇐	art(0,S)	(35) instantiates (23)	(37)
np(0,S)	⇐	n(0,S)	(24) reduces (35)	(38)
np(0,1)			(27) reduces (38)	(39)
s(0,5)	⇐	vp(1,5)	(39) reduces (34)	(40)
vp(1,5)	⇐	v(1,S1) & np(S1,5)	(40) instantiates (25)	(41)
det(0,S)	⇐	gen(1,5)	(39) reduces (36)	(42)
det(0,2)			(28) reduces (42)	(43)
np(0,S)	⇐	n(2,5)	(43) reduces (35)	(44)

EXAMPLE DEDUCTION PROOF (1)

s(S0,S)	⇐	np(S0,S1) & vp(S1,S)		(20)
np(S0,S)	⇐	det(S0,S1) & n(S1,S)		(21)
det(S0,S)	⇐	np(S0,S1) & gen(S1,S)		(22)
det(S0,S)	⇐	art(S0,S)		(23)
det(S0,S)				(24)
vp(S0,S)	⇐	v(S0,S1) & np(S1,S)		(25)
ans	⇐	s(0,5)	goal statement	(33)
s(0,5)	⇐	np(0,S1) & vp(S1,5)	(33) instantiates (20)	(34)
np(0,S)	⇐	det(0,S1) & n(S1,S)	(34) instantiates (21)	(35)
det(0,S)	⇐	np(0,S1) & gen(S1,S)	(35) instantiates (22)	(36)
det(0,S)	⇐	art(0,S)	(35) instantiates (23)	(37)
np(0,S)	⇐	n(0,S)	(24) reduces (35)	(38)
np(0,1)			(27) reduces (38)	(39)
s(0,5)	⇐	vp(1,5)	(39) reduces (34)	(40)
vp(1,5)	⇐	v(1,S1) & np(S1,5)	(40) instantiates (25)	(41)
det(0,S)	⇐	gen(1,5)	(39) reduces (36)	(42)
det(0,2)			(28) reduces (42)	(43)
np(0,S)	⇐	n(2,5)	(43) reduces (35)	(44)
np(0,3)			(29) reduces (44)	(45)

EXAMPLE DEDUCTION PROOF (1)

s(S0,S)	⇐	np(S0,S1) & vp(S1,S)		(20)
np(S0,S)	⇐	det(S0,S1) & n(S1,S)		(21)
det(S0,S)	⇐	np(S0,S1) & gen(S1,S)		(22)
det(S0,S)	⇐	art(S0,S)		(23)
det(S0,S)				(24)
vp(S0,S)	⇐	v(S0,S1) & np(S1,S)		(25)
ans	⇐	s(0,5)	goal statement	(33)
s(0,5)	⇐	np(0,S1) & vp(S1,5)	(33) instantiates (20)	(34)
np(0,S)	⇐	det(0,S1) & n(S1,S)	(34) instantiates (21)	(35)
det(0,S)	⇐	np(0,S1) & gen(S1,S)	(35) instantiates (22)	(36)
det(0,S)	⇐	art(0,S)	(35) instantiates (23)	(37)
np(0,S)	⇐	n(0,S)	(24) reduces (35)	(38)
np(0,1)			(27) reduces (38)	(39)
s(0,5)	⇐	vp(1,5)	(39) reduces (34)	(40)
vp(1,5)	⇐	v(1,S1) & np(S1,5)	(40) instantiates (25)	(41)
det(0,S)	⇐	gen(1,5)	(39) reduces (36)	(42)
det(0,2)			(28) reduces (42)	(43)
np(0,S)	⇐	n(2,5)	(43) reduces (35)	(44)
np(0,3)			(29) reduces (44)	(45)
s(0,5)	⇐	vp(3,5)	(45) reduces (34)	(46)

EXAMPLE DEDUCTION PROOF (2)

s(S0,S)	**⇐**	**np(S0,S1) & vp(S1,S)**		**(20)**
np(S0,S)	**⇐**	**det(S0,S1) & n(S1,S)**		**(21)**
det(S0,S)	**⇐**	**np(S0,S1) & gen(S1,S)**		**(22)**
det(S0,S)	**⇐**	**art(S0,S)**		**(23)**
det(S0,S)				**(24)**
vp(S0,S)	**⇐**	**v(S0,S1) & np(S1,S)**		**(25)**
det(0,S)	**⇐**	**gen(3,S)**	**(45) reduces (36)**	**(47)**

EXAMPLE DEDUCTION PROOF (2)

s(S0,S)	**⇐**	**np(S0,S1) & vp(S1,S)**		**(20)**
np(S0,S)	**⇐**	**det(S0,S1) & n(S1,S)**		**(21)**
det(S0,S)	**⇐**	**np(S0,S1) & gen(S1,S)**		**(22)**
det(S0,S)	**⇐**	**art(S0,S)**		**(23)**
det(S0,S)				**(24)**
vp(S0,S)	**⇐**	**v(S0,S1) & np(S1,S)**		**(25)**
det(0,S)	**⇐**	**gen(3,S)**	**(45) reduces (36)**	**(47)**
vp(3,5)	**⇐**	**v(3,S1) & np(S1,5)**	**(46) instantiates (25)**	**(48)**

EXAMPLE DEDUCTION PROOF (2)

s(S0,S)	⇐	np(S0,S1) & vp(S1,S)		(20)
np(S0,S)	⇐	det(S0,S1) & n(S1,S)		(21)
det(S0,S)	⇐	np(S0,S1) & gen(S1,S)		(22)
det(S0,S)	⇐	art(S0,S)		(23)
det(S0,S)				(24)
vp(S0,S)	⇐	v(S0,S1) & np(S1,S)		(25)
det(0,S)	⇐	gen(3,S)	(45) reduces (36)	(47)
vp(3,5)	⇐	v(3,S1) & np(S1,5)	(46) instantiates (25)	(48)
vp(3,5)	⇐	np(4,5)	(30) reduces (48)	(49)

EXAMPLE DEDUCTION PROOF (2)

s(S0,S)	**⇐**	**np(S0,S1) & vp(S1,S)**		**(20)**
np(S0,S)	**⇐**	**det(S0,S1) & n(S1,S)**		**(21)**
det(S0,S)	**⇐**	**np(S0,S1) & gen(S1,S)**		**(22)**
det(S0,S)	**⇐**	**art(S0,S)**		**(23)**
det(S0,S)				**(24)**
vp(S0,S)	**⇐**	**v(S0,S1) & np(S1,S)**		**(25)**
det(0,S)	**⇐**	**gen(3,S)**	**(45) reduces (36)**	**(47)**
vp(3,5)	**⇐**	**v(3,S1) & np(S1,5)**	**(46) instantiates (25)**	**(48)**
vp(3,5)	**⇐**	**np(4,5)**	**(30) reduces (48)**	**(49)**
np(4,5)	**⇐**	**det(4,S1) & n(S1,5)**	**(49) instantiates (21)**	**(50)**

EXAMPLE DEDUCTION PROOF (2)

s(S0,S)	⇐	np(S0,S1) & vp(S1,S)		(20)
np(S0,S)	⇐	det(S0,S1) & n(S1,S)		(21)
det(S0,S)	⇐	np(S0,S1) & gen(S1,S)		(22)
det(S0,S)	⇐	art(S0,S)		(23)
det(S0,S)				(24)
vp(S0,S)	⇐	v(S0,S1) & np(S1,S)		(25)
det(0,S)	⇐	gen(3,S)	(45) reduces (36)	(47)
vp(3,5)	⇐	v(3,S1) & np(S1,5)	(46) instantiates (25)	(48)
vp(3,5)	⇐	np(4,5)	(30) reduces (48)	(49)
np(4,5)	⇐	det(4,S1) & n(S1,5)	(49) instantiates (21)	(50)
det(4,S)	⇐	np(4,S1) & gen(S1,S)	(50) instantiates (22)	(51)

EXAMPLE DEDUCTION PROOF (2)

s(S0,S)	⇐	np(S0,S1) & vp(S1,S)		(20)
np(S0,S)	⇐	det(S0,S1) & n(S1,S)		(21)
det(S0,S)	⇐	np(S0,S1) & gen(S1,S)		(22)
det(S0,S)	⇐	art(S0,S)		(23)
det(S0,S)				(24)
vp(S0,S)	⇐	v(S0,S1) & np(S1,S)		(25)
det(0,S)	⇐	gen(3,S)	(45) reduces (36)	(47)
vp(3,5)	⇐	v(3,S1) & np(S1,5)	(46) instantiates (25)	(48)
vp(3,5)	⇐	np(4,5)	(30) reduces (48)	(49)
np(4,5)	⇐	det(4,S1) & n(S1,5)	(49) instantiates (21)	(50)
det(4,S)	⇐	np(4,S1) & gen(S1,S)	(50) instantiates (22)	(51)
det(4,S)	⇐	art(4,S)	(50) instantiates (23)	(52)

EXAMPLE DEDUCTION PROOF (2)

s(S0,S)	⇐	np(S0,S1) & vp(S1,S)		(20)
np(S0,S)	⇐	det(S0,S1) & n(S1,S)		(21)
det(S0,S)	⇐	np(S0,S1) & gen(S1,S)		(22)
det(S0,S)	⇐	art(S0,S)		(23)
det(S0,S)				(24)
vp(S0,S)	⇐	v(S0,S1) & np(S1,S)		(25)
det(0,S)	⇐	gen(3,S)	(45) reduces (36)	(47)
vp(3,5)	⇐	v(3,S1) & np(S1,5)	(46) instantiates (25)	(48)
vp(3,5)	⇐	np(4,5)	(30) reduces (48)	(49)
np(4,5)	⇐	det(4,S1) & n(S1,5)	(49) instantiates (21)	(50)
det(4,S)	⇐	np(4,S1) & gen(S1,S)	(50) instantiates (22)	(51)
det(4,S)	⇐	art(4,S)	(50) instantiates (23)	(52)
np(4,S)	⇐	det(4,S1) & n(S1,S)	(51) instantiates (21)	(53)

EXAMPLE DEDUCTION PROOF (2)

s(S0,S)	⇐	np(S0,S1) & vp(S1,S)		(20)
np(S0,S)	⇐	det(S0,S1) & n(S1,S)		(21)
det(S0,S)	⇐	np(S0,S1) & gen(S1,S)		(22)
det(S0,S)	⇐	art(S0,S)		(23)
det(S0,S)				(24)
vp(S0,S)	⇐	v(S0,S1) & np(S1,S)		(25)
det(0,S)	⇐	gen(3,S)	(45) reduces (36)	(47)
vp(3,5)	⇐	v(3,S1) & np(S1,5)	(46) instantiates (25)	(48)
vp(3,5)	⇐	np(4,5)	(30) reduces (48)	(49)
np(4,5)	⇐	det(4,S1) & n(S1,5)	(49) instantiates (21)	(50)
det(4,S)	⇐	np(4,S1) & gen(S1,S)	(50) instantiates (22)	(51)
det(4,S)	⇐	art(4,S)	(50) instantiates (23)	(52)
np(4,S)	⇐	det(4,S1) & n(S1,S)	(51) instantiates (21)	(53)
np(4,5)	⇐	n(4,5)	(24) reduces (50)	(54)

EXAMPLE DEDUCTION PROOF (2)

s(S0,S)	⇐	np(S0,S1) & vp(S1,S)		(20)
np(S0,S)	⇐	det(S0,S1) & n(S1,S)		(21)
det(S0,S)	⇐	np(S0,S1) & gen(S1,S)		(22)
det(S0,S)	⇐	art(S0,S)		(23)
det(S0,S)				(24)
vp(S0,S)	⇐	v(S0,S1) & np(S1,S)		(25)
det(0,S)	⇐	gen(3,S)	(45) reduces (36)	(47)
vp(3,5)	⇐	v(3,S1) & np(S1,5)	(46) instantiates (25)	(48)
vp(3,5)	⇐	np(4,5)	(30) reduces (48)	(49)
np(4,5)	⇐	det(4,S1) & n(S1,5)	(49) instantiates (21)	(50)
det(4,S)	⇐	np(4,S1) & gen(S1,S)	(50) instantiates (22)	(51)
det(4,S)	⇐	art(4,S)	(50) instantiates (23)	(52)
np(4,S)	⇐	det(4,S1) & n(S1,S)	(51) instantiates (21)	(53)
np(4,5)	⇐	n(4,5)	(24) reduces (50)	(54)
np(4,S)	⇐	n(4,S)	(24) reduces (53)	(55)

EXAMPLE DEDUCTION PROOF (2)

s(S0,S)	⇐	np(S0,S1) & vp(S1,S)		(20)
np(S0,S)	⇐	det(S0,S1) & n(S1,S)		(21)
det(S0,S)	⇐	np(S0,S1) & gen(S1,S)		(22)
det(S0,S)	⇐	art(S0,S)		(23)
det(S0,S)				(24)
vp(S0,S)	⇐	v(S0,S1) & np(S1,S)		(25)
det(0,S)	⇐	gen(3,S)	(45) reduces (36)	(47)
vp(3,5)	⇐	v(3,S1) & np(S1,5)	(46) instantiates (25)	(48)
vp(3,5)	⇐	np(4,5)	(30) reduces (48)	(49)
np(4,5)	⇐	det(4,S1) & n(S1,5)	(49) instantiates (21)	(50)
det(4,S)	⇐	np(4,S1) & gen(S1,S)	(50) instantiates (22)	(51)
det(4,S)	⇐	art(4,S)	(50) instantiates (23)	(52)
np(4,S)	⇐	det(4,S1) & n(S1,S)	(51) instantiates (21)	(53)
np(4,5)	⇐	n(4,5)	(24) reduces (50)	(54)
np(4,S)	⇐	n(4,S)	(24) reduces (53)	(55)
np(4,5)			(31) reduces (54)	(56)

EXAMPLE DEDUCTION PROOF (2)

s(S0,S)	⇐	np(S0,S1) & vp(S1,S)		(20)
np(S0,S)	⇐	det(S0,S1) & n(S1,S)		(21)
det(S0,S)	⇐	np(S0,S1) & gen(S1,S)		(22)
det(S0,S)	⇐	art(S0,S)		(23)
det(S0,S)				(24)
vp(S0,S)	⇐	v(S0,S1) & np(S1,S)		(25)
det(0,S)	⇐	gen(3,S)	(45) reduces (36)	(47)
vp(3,5)	⇐	v(3,S1) & np(S1,5)	(46) instantiates (25)	(48)
vp(3,5)	⇐	np(4,5)	(30) reduces (48)	(49)
np(4,5)	⇐	det(4,S1) & n(S1,5)	(49) instantiates (21)	(50)
det(4,S)	⇐	np(4,S1) & gen(S1,S)	(50) instantiates (22)	(51)
det(4,S)	⇐	art(4,S)	(50) instantiates (23)	(52)
np(4,S)	⇐	det(4,S1) & n(S1,S)	(51) instantiates (21)	(53)
np(4,5)	⇐	n(4,5)	(24) reduces (50)	(54)
np(4,S)	⇐	n(4,S)	(24) reduces (53)	(55)
np(4,5)			(31) reduces (54)	(56)
vp(3,5)			(56) reduces (49)	(57)

EXAMPLE DEDUCTION PROOF (2)

s(S0,S)	⇐	np(S0,S1) & vp(S1,S)		(20)
np(S0,S)	⇐	det(S0,S1) & n(S1,S)		(21)
det(S0,S)	⇐	np(S0,S1) & gen(S1,S)		(22)
det(S0,S)	⇐	art(S0,S)		(23)
det(S0,S)				(24)
vp(S0,S)	⇐	v(S0,S1) & np(S1,S)		(25)
det(0,S)	⇐	gen(3,S)	(45) reduces (36)	(47)
vp(3,5)	⇐	v(3,S1) & np(S1,5)	(46) instantiates (25)	(48)
vp(3,5)	⇐	np(4,5)	(30) reduces (48)	(49)
np(4,5)	⇐	det(4,S1) & n(S1,5)	(49) instantiates (21)	(50)
det(4,S)	⇐	np(4,S1) & gen(S1,S)	(50) instantiates (22)	(51)
det(4,S)	⇐	art(4,S)	(50) instantiates (23)	(52)
np(4,S)	⇐	det(4,S1) & n(S1,S)	(51) instantiates (21)	(53)
np(4,5)	⇐	n(4,5)	(24) reduces (50)	(54)
np(4,S)	⇐	n(4,S)	(24) reduces (53)	(55)
np(4,5)			(31) reduces (54)	(56)
vp(3,5)			(56) reduces (49)	(57)
det(4,5)	⇐	gen(5,S)	(56) reduces (51)	(58)

EXAMPLE DEDUCTION PROOF (2)

s(S0,S)	⇐	np(S0,S1) & vp(S1,S)	(20)
np(S0,S)	⇐	det(S0,S1) & n(S1,S)	(21)
det(S0,S)	⇐	np(S0,S1) & gen(S1,S)	(22)
det(S0,S)	⇐	art(S0,S)	(23)
det(S0,S)			(24)
vp(S0,S)	⇐	v(S0,S1) & np(S1,S)	(25)

det(0,S)	⇐	gen(3,S)	(45) reduces (36)	(47)
vp(3,5)	⇐	v(3,S1) & np(S1,5)	(46) instantiates (25)	(48)
vp(3,5)	⇐	np(4,5)	(30) reduces (48)	(49)
np(4,5)	⇐	det(4,S1) & n(S1,5)	(49) instantiates (21)	(50)
det(4,S)	⇐	np(4,S1) & gen(S1,S)	(50) instantiates (22)	(51)
det(4,S)	⇐	art(4,S)	(50) instantiates (23)	(52)
np(4,S)	⇐	det(4,S1) & n(S1,S)	(51) instantiates (21)	(53)
np(4,5)	⇐	n(4,5)	(24) reduces (50)	(54)
np(4,S)	⇐	n(4,S)	(24) reduces (53)	(55)
np(4,5)			(31) reduces (54)	(56)
vp(3,5)			(56) reduces (49)	(57)
det(4,5)	⇐	gen(5,S)	(56) reduces (51)	(58)
s(0,5)			(57) reduces (46)	(59)

EXAMPLE DEDUCTION PROOF (2)

s(S0,S)	⇐	np(S0,S1) & vp(S1,S)		(20)
np(S0,S)	⇐	det(S0,S1) & n(S1,S)		(21)
det(S0,S)	⇐	np(S0,S1) & gen(S1,S)		(22)
det(S0,S)	⇐	art(S0,S)		(23)
det(S0,S)				(24)
vp(S0,S)	⇐	v(S0,S1) & np(S1,S)		(25)
det(0,S)	⇐	gen(3,S)	(45) reduces (36)	(47)
vp(3,5)	⇐	v(3,S1) & np(S1,5)	(46) instantiates (25)	(48)
vp(3,5)	⇐	np(4,5)	(30) reduces (48)	(49)
np(4,5)	⇐	det(4,S1) & n(S1,5)	(49) instantiates (21)	(50)
det(4,S)	⇐	np(4,S1) & gen(S1,S)	(50) instantiates (22)	(51)
det(4,S)	⇐	art(4,S)	(50) instantiates (23)	(52)
np(4,S)	⇐	det(4,S1) & n(S1,S)	(51) instantiates (21)	(53)
np(4,5)	⇐	n(4,5)	(24) reduces (50)	(54)
np(4,S)	⇐	n(4,S)	(24) reduces (53)	(55)
np(4,5)			(31) reduces (54)	(56)
vp(3,5)			(56) reduces (49)	(57)
det(4,5)	⇐	gen(5,S)	(56) reduces (51)	(58)
s(0,5)			(57) reduces (46)	(59)
ans			(59) reduces (33)	(60)

PRESENTING DEDUCTIVE REASONING

Candidates for reasoning steps

- **Resolution unintuitive**
- **Natural deduction too detailed** **systematic transformations between them**
- **Assertion level as application of axioms**

Modifications

- **Some sorts of logical consequences preferably conveyed implicitly through discourse context and default expectations (e.g., 'direct' instantiations)**
- **Divergent human performance in comprehending deductive syllogisms**
  - **91% correct conclusions for modus ponens**
  - **64% for modus tollens,**
  - **48% for affirmative disjunction,**
  - **30% for negative disjunction**
- **Logically redundant pieces of information reintroduced to support the addressee's attention in hard inferences**

Direct proof verbalizations communicatively inadequate!

EXAMPLE 1

Verbose and communicatively redundant text

(1) “Let ρ be an equivalence relation.

Therefore we have ρ is reflexive, we have ρ is symmetric, and we have ρ is transitive.

Then we have ρ is symmetric and we have ρ is reflexive.

Then $\forall x: x \rho x$.

Thus we have $h_0 y_0 \rho h_0 y_0$. … “

EXAMPLE 1

Verbose and communicatively redundant text

(1) "Let ρ be an equivalence relation.

Therefore we have ρ is reflexive, we have ρ is symmetric, and we have ρ is transitive.

Then we have ρ is symmetric and we have ρ is reflexive.

Then $\forall x: x \rho x$.

Thus we have $h_0y_0 \rho h_0y_0$. … "

Exploiting

- **domain knowledge (about equivalence relations)**
- **inference capabilities (and-eliminations)**

Concise and communicatively adequate text

(1') "Let be an equivalence relation.

Thus we have $h_0y_0 \rho h_0y_0$. … "

EXAMPLE 2

Verbose, communicatively redundant, poorly focused text

(2) “Let $1 < a$. Since lemma 1.10 holds, $0 < a^{-1}$.
Since $1 < a$ holds and '<' is monotone, $1a^{-1} < aa^{-1}$ holds.
$a^{-1} < aa^{-1}$ because of the unit element of K.
$aa^{-1} = 1$ because of the inverse element of K for $a \neq 0$.
Thus $a^{-1} < 1$.“

EXAMPLE 2

Verbose, communicatively redundant, poorly focused text

(2) "Let $1 < a$. Since lemma 1.10 holds, $0 < a^{-1}$.
Since $1 < a$ holds and '<' is monotone, $1a^{-1} < aa^{-1}$ holds.
$a^{-1} < aa^{-1}$ because of the unit element of K.
$aa^{-1} = 1$ because of the inverse element of K for $a \neq 0$.
Thus $a^{-1} < 1$."

Exploiting

- **domain knowledge (group axioms)**
- **inference capabilities (their place of application)**

Concise and communicatively adequate text

(2') "Let $1 < a$. Since lemma 1.10 holds, $0 < a^{-1}$.
Then $0 < a^{-1} = 1a^{-1} < aa^{-1} = 1$."

EXAMPLE 3

Hardly comprehensible, communicatively inadequate text

(3) **“ Let ρ be a transitive relation and let $\neg$ ($a\ \rho\ b$).**
Let us assume that $c\ \rho\ b$.
Hence we have $\neg$ ($a\ \rho\ c$).“

EXAMPLE 3

Hardly comprehensible, communicatively inadequate text

(3) “ Let ρ be a transitive relation and let ¬ ($a\ \rho\ b$).
Let us assume that $c\ \rho\ b$.
Hence we have ¬ ($a\ \rho\ c$).“

Taking into account

- **human memory limitations**
  (performing and chaining inferences)
- **degrees of involvedness of an inference**
  (modus tollens + or-elimination)

Sufficiently explicit and communicatively adequate text

(3') “Let ρ be a transitive relation and let ¬ ($a\ \rho\ b$).
Let us assume that $c\ \rho\ b$.
Since ρ is transitive, ¬ ($a\ \rho\ b$) implies that ¬ ($a\ \rho\ c$) or ¬ ($c\ \rho\ b$) holds.
Since we have ¬ ($a\ \rho\ b$) and $c\ \rho\ b$, ¬ ($a\ \rho\ c$) follows.“

A PROBLEM – INTERLEAVING SUBSTRUCTURES

Structural discrepancies (lifting non-elementary text spans)

Too many players hit an acceptable shot,

then stand around admiring it and

wind up losing the point.

There is no time in an action game like tennis to applaud yourself and

still get in position for the next shot.

And you always have to assume there will be a next shot.

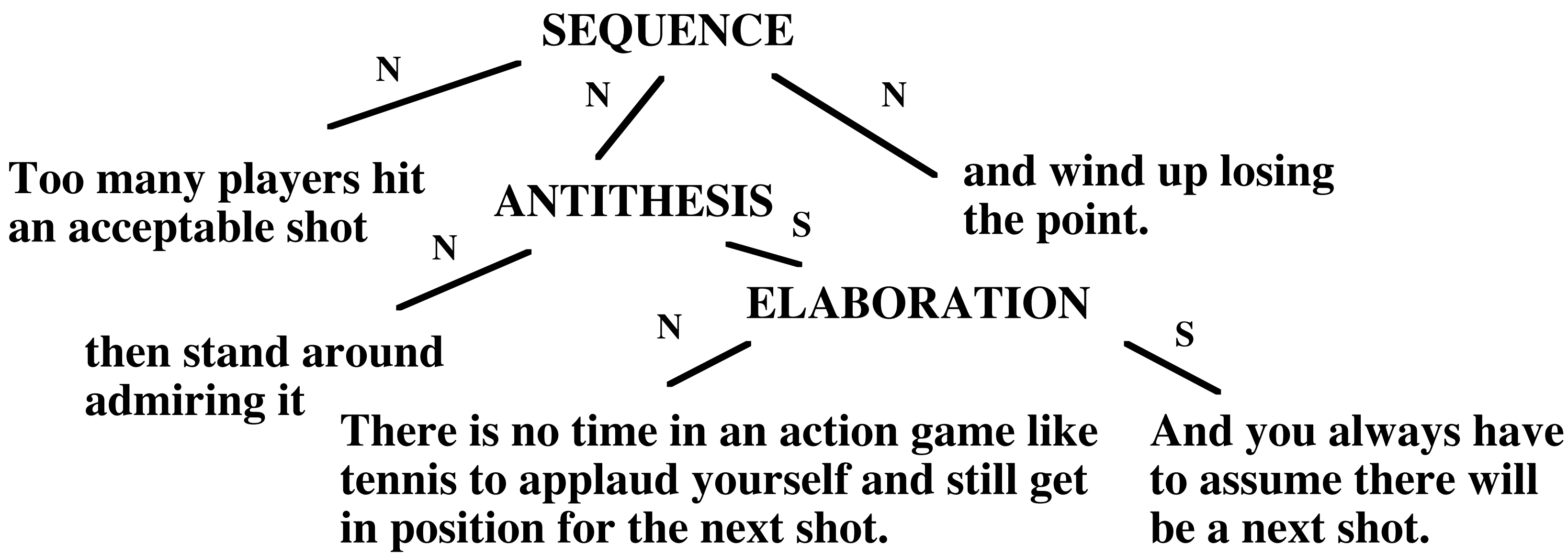
[Mann, Thompson 1987b]

Challenges

- **Inferring the precise scope of arguments – contrasts projection (in analysis)**
- **Restructuring argumentative structures presesrving sequences (in generation)**

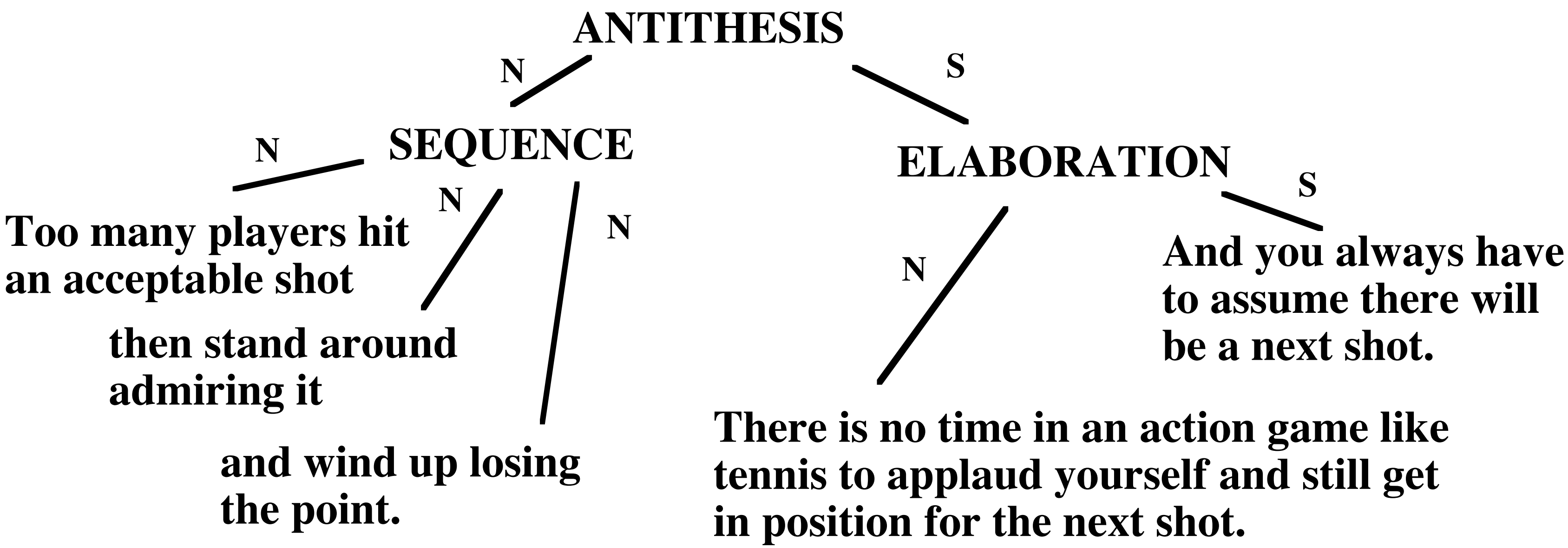
INTERLEAVING SUBSTRUCTURES – EXAMPLE (1)

The semantically "precise" variant



INTERLEAVING SUBSTRUCTURES – EXAMPLE (2)

The rhetorically adequate variant



EVIDENCE FOR BUILDING EXPLANATIONS

Significant discrepancies between proof content and explanation content

Some proof portions perceived as redundant

Some proof steps perceived as involved

Issues to be addressed

Examine discrepancies and their reasons

Develop techniques/procedures that bridge these discrepancies

REPRESENTATION CONCEPTS

Machine-oriented representations

- **Uniformity**
- **Explicitness**
- **Technical reference**

Human-oriented presentations

- **Diversification**
- **Selectivity**
- **Conceptual/linguistic references**

Examples

- **Chess endgame databases – regularities not captured, hardly understood**
- **Expert system presentations**

Commonality-exploiting and rhetorically-oriented recasting

Significant performance improvement in a tutoring system [Di Eugenio et al. 2005]

EXPLICITNESS VERSUS SELECTIVITY

Some sorts of logical consequences preferably conveyed implicitly

through discourse context and default expectations (e.g., 'direct' instantiations)
[Thüring, Wender 1985]

Modus ponens communicated as *Modus brevis*
[Sadock 1977], [Cohen 1987]

Some kinds of "easy" inferable consequences

- **Taxonomic inferences (category memberships)**
- **Normal consequences of actions**
- **Contextually suitable instantiation of rules/regularities mentioned**
- **Responsible causes if sufficiently salient**

USE IN TUTORING ENVIRONMENTS

Building "master" proofs

Deductive reasoning normally done as resolution-based proofs

Transformations to human-oriented representations prior to presentation

Methods for complete proofs inadequate for incremental proof construction

(machine-found proofs may be unintuitive, incrementality, proof step checking)

Semi-automated proof planning

Proof alternatives built off-line on a human-oriented representation level

Incremental proof construction

Matching partial proof step specifications with prefabricated representations

Identification of referred proofs steps resp. subproofs (may be ambiguous)

Verification yields completion of specifications (including direction of reasoning)

OBSTACLES TO EXPLANATIONS

Emphasis in problem-solving on efficiency

Aiming at refutation rather than construction

Disregarding reasoning paths not contributing to a solution

Some reasoning mechanisms widely "explanation-resistent"

(e.g., constraints, neural nets)

Combination of methods

Problem-solving with heterogeneous methods

Dependency on request types (pure problem-solving versus beyond)