CS313K: Logic, Sets, and Functions

J Strother Moore
Department of Computer Sciences
University of Texas at Austin

Lecture 11 – Chap 4 (4.1, 4.2)

Motivation for Factoring

Suppose you wish to prove some formula like:

Goal:

$$(\alpha_1 \ \wedge \ \ldots \ \beta \ \ldots \ \wedge \ \alpha_k) \ \rightarrow \ \gamma$$

and you wish to rewrite `(rev (rev a))` which occurs in β. You wish to use the theorem:

`(true-listp x)` $\rightarrow$ `(rev (rev x))` $=$ `x`

What can you assume about `a`?

Factor $Goal$ into $Goal'$ so that β is in the conclusion of $Goal'$. Then you can assume the hypothesis of $Goal'$.

By defining "factoring" we make it possible to prove $Goal$ by repeatedly rewriting anywhere in the formula without explicitly having to rearrange the formula as part of the proof.

Summaries of the Rules of Inference

To prove ψ using:

- **Tautology**: find a tautology and instantiate it to get ψ.

- **Rewrite**:
  - put ψ into the form $\psi_h \rightarrow (\ldots \alpha' \ldots)$ where α' is the term you want to change,
  - pick some theorem in the form $\phi_h \rightarrow (\alpha = \beta)$ or $\phi_h \rightarrow (\alpha \leftrightarrow \beta)$,
  - match α with α' so that $\alpha/\sigma = \alpha'$,
  - relieve the hypotheses by proving $\psi_h \rightarrow \phi_h/\sigma$,
  - make sure the equivalence ("$=$" or "$\leftrightarrow$") is ok,
  - replace α' with β/σ.

- **Hypothesis**:
  - pick a term α' in the conclusion of ψ to rewrite with some hypothesis $(\alpha = \beta)$ or $(\alpha \leftrightarrow \beta)$,
  - make sure the equivalence ("$=$" or "$\leftrightarrow$") is ok,
  - replace α with β.

- **Cases**:
  - pick an exhaustive set of terms $(\phi'_1 \vee \ldots \vee \phi'_k)$
  - for each, prove $\phi'_i \rightarrow \psi$.

- **Constant Expansion (Variant 1)**:
  - pick a list constant `'(`α `...)` in ψ
  - replace it by `(cons '`α `'...)`

- **Constant Expansion (Variant 2)**:
  - pick a non-0 natural n in ψ
  - replace it by `(+ 1` n'`)`, where n' is $n - 1$.

- **Computation**:
  - pick a function call `(`f c_1 `...` c_n`)` in ψ, where the c_i are all constants
  - run f on those inputs to get v
  - replace `(`f c_1 `...` c_n`)` by `'`v