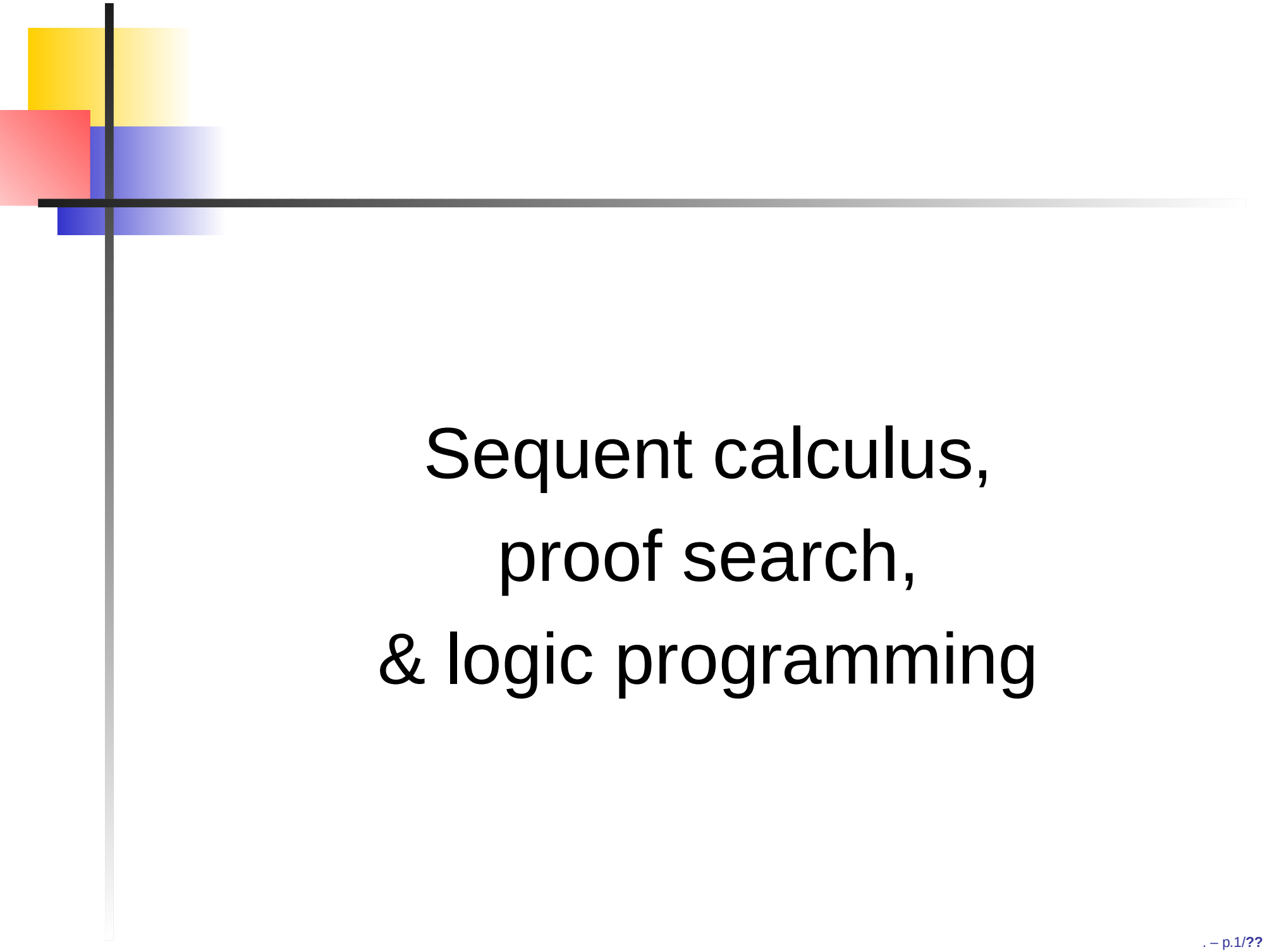

Sequent calculus, proof search, & logic programming

Deductive vs. reductive inference

- **Deductive inference** proceeds from premises to a conclusion:

$$\frac{\text{premise}_1 \ldots \text{premise}_n}{\text{conclusion}} \Downarrow$$

- **Reductive inference** proceeds **backwards** from a putative conclusion or **goal sequent** to sufficient sets of premises:

$$\frac{\text{premise}_1 \ldots \text{premise}_n}{\text{putative conclusion}} \Uparrow$$

Proof search

- We call reductive inference **proof search**.
- There can be many choices for reducing a goal sequent. E.g. the goal sequent below could be reduced in five ways.

$$A \wedge B, C \rightarrow (D \rightarrow E), (A \wedge C) \rightarrow E \vdash E \vee B, B \rightarrow D$$

- So we have a **search space**: all possible attempts at reducing the goal sequent.

Opting for additive rules

For proof search, additive rules are better than multiplicative rules. For example, given the goal sequent

$$\Gamma \vdash A \wedge B, \Delta,$$

applying additive $R\wedge$ backwards yields

$$\Gamma \vdash A, \Delta \qquad \Gamma \vdash B, \Delta,$$

while applying multiplicative $R\wedge$ yields

$$\Gamma_1 \vdash A, \Delta_1 \qquad \Gamma_2 \vdash B, \Delta_2$$

for **any** splitting of $\Gamma = \Gamma_1, \Gamma_2$ and $\Delta = \Delta_1, \Delta_2$. Evidently, we better avoid having to choose such a splitting.

Avoiding cut

The cut rule is bad for proof search, because it violates the subformula property. E.g., applying (additive) cut backwards to

$$\Gamma \vdash \Delta$$

yields the new goal sequents below:

$$\Gamma \vdash A, \Delta \qquad \Gamma, A \vdash \Delta.$$

Evidently, we better avoid having to guess A. Fortunately, owing to the cut-elimination theorem, we can prove everything without cut!

Search space still too big

- But even without cut and with only additive rules, the search space turns out too big for realistic proof search.
- The reason is the number of choices for picking the principle formula. E.g. recall that

$$A \wedge B, C \rightarrow (D \rightarrow E), (A \wedge C) \rightarrow E \vdash E \vee B, B \rightarrow D$$

provides five choices!

Towards logic programming

Logic programming limits the search-space by focusing on sequents $\Gamma \vdash A$ with a single succedent A. We write

$$\Gamma \ ?\!- \ A$$

- A is called the **goal formula** of simply **goal**.
- Γ is called the **program**, because it provides the instructions for proving A, as we shall see.
- $?\!-$ stands for the **inference engine**. (E.g. Prolog).

Sequent calculus for proof search

- Logic programming is best discussed in the context of an additive, cut-free, single-succendent sequent calculus.
- It helps to consider contexts Γ to be sets of formulæ, not lists.
- This corresponds to making the rules LE, RE, LC, RC implicit.
- The rule WR no longer makes sense, because of single succedents.
- WL is not an inference rule, but it is derivable.

The "minimal sequent calculus"

$$\frac{}{\Gamma, A \vdash A}\, Ax$$

$$\frac{\Gamma, A, B \vdash C}{\Gamma, A \wedge B \vdash C}\, L\wedge \qquad \frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\, R\wedge$$

$$\frac{\Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma, A \vee B \vdash C}\, L\vee \qquad \frac{\Gamma \vdash A_i}{\Gamma \vdash A_1 \vee A_2}\, (i = 1, 2) R\vee$$

$$\frac{\Gamma \vdash A \quad \Gamma, B \vdash C}{\Gamma, A \to B \vdash C}\, L\to \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\, R\to$$

$$\frac{\Gamma, A[t/x] \vdash B}{\Gamma, \forall x.A \vdash B}\, L\forall \qquad \frac{\Gamma \vdash A}{\Gamma \vdash \forall x.A}\, R\forall$$

$$\frac{\Gamma, A \vdash B}{\Gamma, \exists x.A \vdash B}\, L\exists \qquad \frac{\Gamma \vdash A[t/x]}{\Gamma \vdash \exists x.A}\, R\exists$$

Completeness?

- This calculus is not complete w.r.t. the usual semantics of predicate logic!
- However, only two rules are missing:

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash A}\ \text{ex falso quodlibet} \qquad \frac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash A}\ \text{RAA.}$$

- In fact, RAA implies EFQ.
- The calculus without these two rules is for **minimal logic**.
- The calculus without RAA but with EFQ is for **intuitionistic** logic. More about this later.

Uniform proofs

- Logic programming constrains the search space outlined by the minimal sequent calculus even more.
- This can be explained elegantly in terms of **uniform proofs** (Dale Miller et. al.).
- The idea is that the goal is taken to pieces (by right rules) as long as possible; left rules are applied only when the goal is atomic.

Uniform proofs: definition

Definition. A proof in the minimal sequent calculus is **uniform** if every sequent $\Gamma \vdash A$ with non-atomic succedent A is obtained from a right rule.

Completeness?

- Problem: uniform proofs are not even complete for the minimal sequent calculus (consider e.g. $p \vee q \vdash p \vee q$).
- Solution: characterize a class of sequents for which uniform proofs **are** complete.

Hereditarily Harrop sequents

Definition. A **Hereditarily Harrop** sequent is of the form

$$D_1, \ldots, D_n \vdash G,$$

where the D's (**definite clauses**) and G (**goal**) obey the grammar

$$D ::= \bot | p | G \to p | G \to \bot | \forall x.D | D_1 \wedge D_2$$
$$G ::= \bot | p | G_1 \wedge G_2 | G_1 \vee G_2 | \exists x.G | D \to G.$$

Prolog as a special case of HH sequents

```
mortal(X) :- human(X).
featherless(socrates).
bipedal(socrates).
animal(socrates).
human(X) := featherless(X),
            bipedal(X), animal(X).
```

This corresponds to the following set Γ of definite clauses (note the $\forall$-quantifier):

$$\forall x.human(x) \rightarrow mortal(x), featherless(socrates),$$

$$bipedal(socrates), animal(socrates),$$

$$\forall x.featherless(x) \wedge bipedal(x) \wedge animal(x) \rightarrow human(x)$$

Prolog as a special case of HH sequents

By contrast, a Prolog query, e.g.

```
?- featherless(X),bipedal(X),animal(X)
```

corresponds to the goal (note the $\exists$-quantifier):

$$\exists x.featherless(x) \wedge bipedal(x) \wedge animal(x)$$

Prolog as a special case of HH sequents

So a query to Prolog program can be considered as a special case of a HH sequent

$$D_1, \ldots, D_n \vdash G$$

where

- each D_i is of the form $\forall x_1. \ldots \forall x_n. (p_1 \wedge \ldots \wedge p_k \rightarrow q)$, where the p_i and q are atomic, and
- G is of the form $\exists x_1. \ldots \exists x_n. r_1 \wedge \ldots \wedge r_m$, where the r_i are atomic.

Lambda-prolog

- The full power of HH sequents is implemented in **Lambda-Prolog**.
- In particular, it allows goals of the form $D \rightarrow G$.
- D can be seen as code to be loaded prior to proving G.

Completeness of uniform proofs

Theorem. Uniform proofs of Hereditarily Harrop sequents are complete w.r.t. minimal predicate logic.

Proof. By re-writing proofs in the minimal sequent calculus into uniform proofs.

Towards resolution

Suppose our goal sequent is

$$\Gamma, \forall x.G(x) \rightarrow p(x) \vdash \exists x.q(x),$$

where p and q are atomic, and there is a term t such that $p(t) = q(t)$. Then, by uniform proof, we can reduce the goal sequent to $\Gamma \vdash G(t)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\Gamma \vdash G(t)} \quad \dfrac{}{\Gamma, p(t) \vdash q(t)}\, Ax}{\Gamma, G(t) \rightarrow p(t) \vdash q(t)}\, L\rightarrow}{\Gamma, \forall x.G(x) \rightarrow p(x) \vdash q(t)}\, L\forall}{\Gamma, \forall x.G(x) \rightarrow p(x) \vdash \exists x.q(x)}\, R\exists$$

Resolution

In other words, to prove

$$D_1, \ldots, D_n \vdash \exists x.q(x),$$

find a D_i of the form

$$D_i = \forall x.G(x) \rightarrow p(x)$$

and a term t such that $p(t) = q(t)$. Then it suffices to prove $D_1, \ldots, D_n \vdash G(t)$. Such a t can be found by **unification**; Prolog essentially works in that way (recall CM20019).

Completeness of resolution

A **resolution proof** is a uniform proof such that existential goals $\exists x.A$ are treated as we just described. (The precise description is a bit more complicated, because A need not be atomic.)

Theorem. Resolution proofs of HH formulæ are complete w.r.t. minimal predicate logic.

Proof. By showing that every uniform proof can be re-written into a resolution proof.