Applied Logic	**Lecture 11: Refinement Logic**
CS 4860 Spring 2009	**Tuesday, February 24, 2009**

Roadmap - where are we?
We have started with formulas and valuations, looked at the truth table method and devidated into SAT solving – automatic procedures, NP completeness and the like.
We then looked at proof calculi for propositional logic that are better suited for proving validity and have the potential to be extended beyond propositional logic. We looked at tableaux as a refutation calculus, proved them consistent and complete for propositional logic, and proved compactness
After that we looked at Gentzen systems that focus more on the idea of a (positive) proof rather than a failing refutation and showed that technically they are more or less isomorphic to tableaux.
Today I am going to introduce one further modification of the calculus that is best suited for interactive proof development, constructive reasoning, and can be extended well to first- and higher order logic and computational type theory.
Once we have that we can actually go beyond propositional logic and look at the more interesting logics.

Gentzen systems as presented in Smullyan's book and Gentzen's original papers, allow a sequent to have multiple conclusions, which sometimes leads to confusing proofs where the evidence for the validity of the formula is not obvious. We will investigate a few of these proofs and then introduce a sequent calculus that does not require multiple succedents. This *refinement logic* has many properties that are interesting for a computer scientists, because it catches the notion of a construction in a very nice and natural way. It is also the logic that we use in our Nuprl system and at some later time I will show you how we can use this system to support and automate the development of formal proofs.

11.1 Oddities of multi-conclusioned Gentzen Systems

The fact that we allow multiple goals in a sequent leads to two kinds of oddities. The first shows up in a sequent proof of Pierce's Law $((P \supset Q) \supset P) \supset P$ that we already proved in the tableau calculus.

```
     ⊢ ((P⊃Q)⊃Q)⊃P        by ⊃R
     (P⊃Q)⊃P ⊢ P          by ⊃L
[1]    ⊢ P,  P⊃Q          by ⊃R
       P ⊢ P, Q           by axiom
[2]    P ⊢ P              by axiom
```

In subgoal [1] we have used the ⊃R rule in an attempt to prove $P \supset Q$ but in the subgoal resulting from that we didn't prove the right hand side Q of the implication but used the left hand side P of the implication to prove the *other conclusion* P that we already had before. In a sense this is a bait-and-switch strategy: when we decompose the conclusion $P \supset Q$ one would expect us to go on proving Q but instead we change our mind after we have been given P as a new assumption and prove a different goal instead. For some people that appears almost like cheating. But according to the rules of the calculus this is perfectly acceptable. And since the calculus is consistent, we have to accept this kind of reasoning as long as we accept truth tables and the semantics of formulas that we discussed in the early lectures of this class.

We get a similar problem when we try to prove the law of excluded middle

```
⊢ P∨∼P          by ∨R
⊢ P,∼P          by ∼R
P ⊢ P           by axiom
```

Again, there is something strange about this proof: we are switching to the other goal once we have decomposed the negation.

The difficulty with these proofs is that switching goals in the middle of a proof doesn't allow us to see the construction of the proof argument (in fact, we cannot construct proper evidence for these proofs). It seems more natural to have proofs that focus on one particular conclusion instead of allowing several ones at the same time which make it possible to change ones mind about what to show in the middle of a proof. The only meaningful way to address this issues is restricting the right hand side to a single conclusion.

The second oddity comes up in the proof of the law of contraposition

```
    ⊢ (P⊃Q) ⊃ (∼Q⊃∼P)          by ⊃R
     P⊃Q ⊢ ∼Q⊃∼P               by ⊃R
     P⊃Q, ∼Q ⊢ ∼P              by ∼R
     P⊃Q, ∼Q, P ⊢               by ⊃L
[1]     ∼Q, P ⊢ P               by axiom
[2]     Q, ∼Q, P ⊢              by ∼L
        Q, P ⊢ Q                by axiom
```

Although this proof is perfectly fine, there are two subgoals with no conclusion at all. What is the meaning of such a goal, considering the fact that we understand a sequent to mean "prove one of the conclusions"? Given that a set of conclusions corresponds to a disjunction of these conclusions, an empty set means that we have to prove false, i.e. that the hypotheses are contradictory.

A solution for that is to introduce a constant `f` to represent falsehood and to consider negation $\sim X$ as abbreviation for $X \supset$`f`. The rules for negation then become redundant, if we add a rule for `f`:

$$H, \mathtt{f} \vdash G$$

It is easy to see that this rule, together with the rules for $\supset$ covers the two rules for negation.

11.2 Single-conclusioned Gentzen Systems

We will now discuss a proof calculus that results from the above considerations and leads to proofs that are entirely constructive. This calculus, which we intend to use in the rest of this course, is called *refinement logic* and results from restricting Gentzen systems to single-conclusioned sequents, adding the constant `f`, and considering negation $\sim X$ as abbreviation for $X \supset$`f` , and adding the constant `f`. Refinement logic calculus is simpler and more focused that multi-conclusioned Gentzen systems, but it puts limits one the methods that we can use to prove formulas valid: there will be no more bait-and-switch.

The rules of refinement logic can be derived from those of Gentzen systems by dropping the extra conclusions, (that is the set G that occurs in each `R`-rule after the turnstyle and before the comma), from each sequent. For most rules this transformation is straightforward. For the $\supset$`L` rule

$\supset$L $\quad H, A \supset B \vdash G$
$\qquad H, \vdash G, \mathrm{A}$
$\qquad H, \mathrm{B} \vdash G$

the transformation requires us to drop the original goal G in the first subgoal and to keep only A (similarly for $\sim$L). The only rule that requires a major modification is the rule $\vee$R, which explicitly generates two conclusions.

$\vee$R $\quad H, \vdash G, A \vee B$
$\qquad H, \vdash G, \mathrm{A}, \mathrm{B}$

Since we cannot generate two conclusions we must choose which of the two disjuncts we want to prove and continue with that one. As a result $\vee$R will have to be replace by two inference rules – one for choosing A, then other for choosing B.

The above modifications of $\supset$L, $\sim$L, and $\vee$R are necessary to maintain a single conclusion in the course of a proof. The resulting rules, however, have become irreversible, as we will show later. Even worse – the proof system will become incomplete. We will only be able to prove formulas that have a computational meaning (i.e. formulas for which one can construct positive evidence instead of just showing that they cannot be refuted). Formulas like $P \vee \sim P$ or Pierce's law become unprovable unless we extend the calculus by a special rule like the law of excluded middle or some equivalent law.

11.3 Core Refinement Logic

	left	right	
andL	$H, \mathrm{A} \wedge \mathrm{B}, H' \vdash \mathrm{G}$ $H, \mathrm{A}, \mathrm{B}, H' \vdash \mathrm{G}$	$H \vdash \mathrm{A} \wedge \mathrm{B}$ $H \vdash \mathrm{A}$ $H \vdash \mathrm{B}$	andR
orL i	$H, \mathrm{A} \vee \mathrm{B}, H' \vdash \mathrm{G}$ $H, \mathrm{A}, H' \vdash \mathrm{G}$ $H, \mathrm{B}, H' \vdash \mathrm{G}$	$H \vdash \mathrm{A} \vee \mathrm{B}$ $H \vdash \mathrm{A}$ $H \vdash \mathrm{A} \vee \mathrm{B}$ $H \vdash \mathrm{B}$	orR1 orR2
impL i	$H, \mathrm{A} \Rightarrow \mathrm{B}, H' \vdash \mathrm{G}$ $H, \mathrm{A} \Rightarrow \mathrm{B}, H' \vdash \mathrm{A}$ $H, H', \mathrm{B} \vdash \mathrm{G}$	$H \vdash \mathrm{A} \Rightarrow \mathrm{B}$ $H, \mathrm{A} \vdash \mathrm{B}$	impR
notL i	$H, \neg\mathrm{A}, H' \vdash \mathrm{G}$ $H, \neg\mathrm{A}, H' \vdash \mathrm{A}$	$H \vdash \neg\mathrm{A}$ $H, \mathrm{A} \vdash \mathrm{f}$	notR
falseL i	$H, \mathrm{f}, H' \vdash \mathrm{G}$		
axiom i	$H, \mathrm{A}, H' \vdash \mathrm{A}$		

Refinement Logic as implemented in Nuprl uses a slightly different notation for logical connectives. Implication is $\Rightarrow$ instead of $\supset$, negation is $\neg$ instead of $\sim$ and often viewed as defined connective instead of a basic one, i.e. $\neg A$ is viewed as abbreviation for $A \Rightarrow \mathrm{f}$. Also, in a computer system we have to use lists of formulas instead of sets, and for the left rule the formula to

be decomposed may be somwhere within that list, so all left rules must provide an index i of the hypothesis to indicate the formula to which the rule shall be applied.

Note that due to the transition from set- to list-notation we have to state explicitly that the `impL` rule has to preserve the implication that is being decomposed. We only need this implication the first subgoal, since we can use its right hand side as assumption in the second subgoal.

Furthermore, the restriction to a single conclusion makes the construction of evidence fully deterministic, which resolves all the problematic issues that came up in our discussion of multi-conclusioned Gentzen systems in lecture 9. The fact that we now have two rules for dealing with disjunctions in the conclusion enables us to construct evidence that clearly states which of the two disjuncts had been proven – it was the left one if we used `orR1` and the right one if we used `orR2`. The ambiguitiy of the evidence of the `axiom` rule has disappeared since there is only one conclusion that can be proven by the given labelled assumption. The ambiguitiy of the evidence of `impL` has disappeared since the only conclusion in the first subgoal is now the left hand side of the implication. The following table shows the evidence constructed by the application of refinement logic rules.

	left		right	
`andL`	$H, \mathrm{A} \wedge \mathrm{B}, H' \vdash \mathrm{G}$	ev = `let` x=(a,b) `in` $g[a,b]$	$H \vdash \mathrm{A} \wedge \mathrm{B}$ ev = (a,b)	`andR`
	$H, a{:}\mathrm{A}, b{:}\mathrm{B}, H' \vdash \mathrm{G}$	ev = $g[a,b]$	$H \vdash \mathrm{A}$ ev = a	
			$H \vdash \mathrm{B}$ ev = b	
`orL` i	$H, \mathrm{A} \vee \mathrm{B}, H' \vdash \mathrm{G}$	ev = `case` x `of` `inl`$(a) \mapsto g_1[a]$ `| inr`$(b) \mapsto g_2[b]$	$H \vdash \mathrm{A} \vee \mathrm{B}$ ev = `inl`(a)	`orR1`
	$H, a{:}\mathrm{A}, H' \vdash \mathrm{G}$	ev = $g_1[a]$	$H \vdash \mathrm{A}$ ev = a	
	$H, b{:}\mathrm{B}, H' \vdash \mathrm{G}$	ev = $g_2[b]$	$H \vdash \mathrm{A} \vee \mathrm{B}$ ev = `inr`(b)	`orR2`
			$H \vdash \mathrm{B}$ ev = `inr`(b)	
`impL` i	$H, f{:}\mathrm{A} \Rightarrow \mathrm{B}, H' \vdash \mathrm{G}$	ev = $g[f(a)/b]$	$H \vdash \mathrm{A} \Rightarrow \mathrm{B}$ ev = `fun` $a \rightarrow b[a]$	`impR`
	$H, f{:}\mathrm{A} \Rightarrow \mathrm{B}, H' \vdash \mathrm{A}$	ev = a	$H, a{:}\mathrm{A} \vdash \mathrm{B}$ ev = $b[a]$	
	$H, H', b{:}\mathrm{B} \vdash \mathrm{G}$	ev = $g[b]$		
`notL` i	$H, n{:}\neg\mathrm{A}, H' \vdash \mathrm{G}$	ev = `any`$[f(a)]$	$H \vdash \neg\mathrm{A}$ ev = `fun` $a \rightarrow f[a]$	`notR`
	$H, n{:}\neg\mathrm{A}, H' \vdash \mathrm{A}$	ev = a	$H, \mathrm{A} \vdash \mathrm{f}$ ev = $f[a]$	
`falseL` i	$H, z{:}\mathrm{f}, H' \vdash \mathrm{G}$	ev = `any`$[z]$		
`axiom` i	$H, a{:}\mathrm{A}, H' \vdash \mathrm{A}$	ev = a		

Because of the restriction of Gentzen rules to single-conclusioned sequents, some of the proofs that we developed in earlier examples will not go through anymore. Consider for instance, the proof of Pierce's law

```
     ⊢ ((P ⇒ Q) ⇒ P) ⇒ P        by impR
     (P ⇒ Q) ⇒ P ⊢ P            by impL
[1]     (P ⇒ Q) ⇒ P ⊢ P ⇒ Q     by impR
        (P ⇒ Q) ⇒ P, P ⊢ Q      by ???
[2]      P ⊢ P                  by axiom
```

There is no way to complete the proof, since we will only get back to the same goal over and over again. As a matter of fact, the formula `((P ⇒ Q) ⇒ P) ⇒ P` cannot be proven at all with the rules

that we described so far. Even worse, we can't even prove the law of excluded middle anymore, since the rules for disjunction on the right force us to make a choice how to proceed.

```
⊢ P∨¬P      by ∨R1                ⊢ P∨¬P      by ∨R2
⊢ P         ??                    ⊢ ¬P        ??
```

In both cases the elimination of a bait-and-switch strategy with multiple conclusions costs us the ability to prove theorems that are known to be true in propositional logic. For these theorems we have a tableau proof, a multi-conclusioned sequent proof, and even a truth table proof – so they must be true.

Q: *What do we do?*

11.4 Add the law of excluded middle and the cut rule

If we want to maintain completeness we must add something to the inference system again to compensate for what we lost due to the restriction to single conclusions. The easiest way to do that is to add one of the formulas that we cannot prove anymore as basic inference rule. One of the most simple of these formulas is the law of excluded middle $P \vee \neg P$, which is considered a fundamental truth of propositional logic (we could equally well use $\neg\neg P \Rightarrow P$). While one may dispute that this is actually a fundamental law of logic because it has no constructive meaning, it *is* a fundamental truth in the propositional logic defined it so far – a boolean valuation of a formula can only be true or false.

For the sake of convenience, we add the law of excluded middle as a rule that allows us to add an instance of this law to the list of assumptions whenever it is needed.

`magic` A $\quad H \vdash \mathrm{G}$ $H, A \vee \neg A \vdash \mathrm{G}$	$H \vdash \mathrm{G}$ $\quad$ `cut` A $H \vdash A$ $H, A \vdash \mathrm{G}$

We call this rule `magic`, because it magically introduces a fact that has nothing to do with the proof so far. In fact, this rule breaks the clean design of Gentzen systems. While all other rules generate only subformulas of the original proof goal as assumption or conclusion of a proof sequent, the `magic` rule may introduce an entirely new formula.[1] This formula has to be given as a parameter to the rule since it cannot be identified as a (sub-)formula of the current sequent. Furthermore, it is not possible to associate this rule with the construction of evidence, because it just states that one of A or $\neg A$ has to be true without providing evidence for that claim. As a matter of fact, it is often impossible to show which of the two is true.

For reasons that will become apparent later, we also add another rule to the calculus. It allows us to state and prove intermediate results and use them as assumption in the rest of the proof. As this rule cuts the proof into smaller segments that are much easier to handle, it is called the `cut` rule. As for the `magic` rule the formula A has to be provided as parameter for `cut`.

[1] Logicians also observe that all other rules deal with a single logical connective while the magic rule combines two.

Let us see how this works out with Pierce's law. What do we need to change to make it work? We can be sure that we need the magic rule somewhere – the only question is how?

Q: *Why did the proof without magic break?*

Because the `impL` rule in the second step took away the conclusion `P` and replaced it by the left hand side of the implication. Later then, when we tried bait-and-switch after decomposing $P \Rightarrow Q$ it didn't work anymore, because we couldn't use the alternative conclusion and were stuck with `Q`. Applying the magic rule means re-introducing that goal before the second step and preserving it somehow in the hypotheses list.

```
    ⊢ ((P⇒Q)⇒Q)⇒P          by impR
    (P⇒Q)⇒P ⊢ P            by magic P
    (P⇒Q)⇒P, P∨¬P ⊢ P      by orL
[1]   (P⇒Q)⇒P, P ⊢ P       by axiom
[2]    (P⇒Q)⇒P, ¬P ⊢ P     by impL
[2.1] (P⇒Q)⇒P, ¬P ⊢ P⇒Q   by impR
      (P⇒Q)⇒P, ¬P, P ⊢ Q   by notL
      (P⇒Q)⇒P, ¬P, P ⊢ P   by axiom
[2.2] ¬P, P ⊢ P            by axiom
```

```
    ⊢ ((P⇒Q)⇒Q)⇒P     by ⇒R
    (P⇒Q)⇒P ⊢ P       by ⇒L
[1]  ⊢ P,  P⇒Q        by ⇒R
      P ⊢ P, Q        by axiom
[2]  P ⊢ P            by axiom
```

As we see, we get our alternative goal back after applying `impR` since we kept its negation in the hypotheses. So when we decomposed $P \Rightarrow Q$, we had a contradictory hypotheses list, used `notL` to move `P` back to the right hand side and then closed the proof.

Compare this to the multi-conclusioned proof on the right. It has the same structure, but we had to do a lot of extra work to fill in the blanks. In fact, we had to do them at the right time. Had we used the magic rule after `impL` in the second step, we wouldn't have been able to complete the proof anymore.

11.5 Discussion

The fact that we were able to complete the proof of Pierce's law but only under considerable efforts raises a few questions about refinement logic.

1. *Is it consistent?*
2. *Is it complete?*
3. *Can a proof search get wedged?*
4. *What kind of logic do we get if drop the* `magic` *rule?*
5. *Is it decidable?*

Given that the inference rules are a result of weakening the rules of Gentzen systems and using only obvious truths as additional rules is very plausible that refinement logic is consistent, but obviously we have to prove that.

Concerning completeness we know already that the calculus is *not* complete without the `magic` rule. But we have to show that adding `magic` and `cut` to the calculus is sufficient to be able to prove everything that we could prove with Gentzen systems.

The question whether a proof search can get wedged is something that didn't come up with analytic tableaux or Gentzen systems. In these calculi *every proof attempt will succeed* if the formula is valid. But the example proof of Pierce's law showed us that some formulas have quite complex proofs in refinement logic even if the Gentzen proof is straightforward. Thus it could very well be the case that we may not always be able to find a proof if we start wrong. This doesn't affect completeness because completeness just says that every valid formula is provable but not that every proof attempt has the chance to succeed.

Although refinement logic without the `magic` rule is incomplete it has the interesting property that every proof provides constructive evidence for the validity of the proven formula. From a computational perspective this is a valuable property so it may be worth investigating what kind of logic is implicitly defined by core refinement logic.

Decidability is another interesting issue that we haven't discussed yet. Is there a computational method to decide whether a formula is valid or not? Even more, can we always find a proof in refinement logic if the formula is valid? If a proof search can get wedged this is not a trivial problem because one has to provide a proof search method that successfully avoids getting stuck.

We will investigate all these questions in the next lecture.