11 Refinement Logic

Gentzen systems, as presented in Smullyan's book and Gentzen's original papers, allow a sequent to have multiple conclusions. Today we will go one step further and introduce a sequent calculus that doesn't use 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 Some 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 $((P\supset Q)\supset P)\supset P$

```
     ⊢ ((P⊃Q)⊃P)⊃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 this case we used the ⊃R rule in an attempt to prove $P\supset Q$ but in the subgoal we used the new assumption P not to prove Q but to prove the other goal P that we already had. For some people that appears like cheating, like a bait-and-switch strategy, or at least weird, although it is perfectly legal since we are using a consistent calculus. So if you accept truth tables, you must also accept this proof.

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, as we're switching to the other goal once we have decomposed the negation. The difficult thing about this kind of proofs is that switching goals in the middle of a proof doesn't allow us to see the construction of the proof argument anymore. It seems more natural to have proofs that focus on one particular conclusion instead of allowing several ones at the same time. So restricting the right hand side to one conclusion appears to be appropriate.

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 ok, 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 an abbreviation for $X \supset$`f`. The rules for negation then become superfluous, but we have to add a rule for `f`: H, `f` $\vdash G$. It is easy to see that this rule, together with the rules for implication covers the two rules for negation.

11.2 Refinement Logic: single-conclusioned Gentzen Systems

The logic that we intend to use in the rest of this course, *refinement logic*, results from restricting Gentzen systems to single-conclusioned sequents, dropping negation, and adding the constant `f`. The resulting calculus is simpler and more focused, but it will limit our flexibility when it comes to actually proving things (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 gefore the comma, from each sequent. This works fine except in the case of disjunction, since the $\vee$R rule explicitly generates two conclusions and we can’t have that anymore. So we need to replace it by two rules, whose application determines which of the two alternatives we intend to follow.

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 $A \Rightarrow$ 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.

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

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

If we restrict the Gentzen rules to single-conclusioned sequents as described above, we will find out that certain proofs do 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. 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, forbidding the bait-and-switch strategy with multiple conclusions costs us the ability to prove theorems that are known to be true in propositional logic. After all – we have a tableau proof, a multi-conclusioned sequent proof, and even a truth table proof for these theorems. So, what do we do?

11.3 Add the law of excluded middle and the cut rule

We must add something to the rule system again that apparently got lost. And one of the assumptions in propositional logic is that the law of excluded middle, which we can't prove anymore, is a fundamental truth of logic. We may dispute that this law is in fact fundamental to logic – we will do so in a minute – but in the propositional logic as we defined it so far, this law *is* a fundamental truth, as a boolean valuation of a formula can only be true or false. So we add the law of excluded middle as an axiom to our rule system.

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 hypotheses list, whenever it is needed. We call this rule `magic`, because it magically introduces a fact that has nothing to do with the proof so far.

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

For reasons that become apparent later, we also add a second rule to the calculus, which 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. In both `magic` and `cut` the formula A has to be provided.

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: $\boxed{\textit{Why did the proof without magic break?}}$

Because the `impL` 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 the bait-and-switch trick 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.

$\vdash ((P \Rightarrow Q) \Rightarrow Q) \Rightarrow P$	by `impR`	$\vdash ((P \Rightarrow Q) \Rightarrow Q) \Rightarrow P$	by $\Rightarrow R$
$(P \Rightarrow Q) \Rightarrow P \vdash P$	by `magic P`		
$(P \Rightarrow Q) \Rightarrow P,\ P \vee \neg P \vdash P$	by `orL`		
[1] $(P \Rightarrow Q) \Rightarrow P,\ P \vdash P$	by `axiom`		
[2] $(P \Rightarrow Q) \Rightarrow P,\ \neg P \vdash P$	by `impL`	$(P \Rightarrow Q) \Rightarrow P \vdash P$	by $\Rightarrow L$
[2.1] $(P \Rightarrow Q) \Rightarrow P,\ \neg P \vdash P \Rightarrow Q$	by `impR`	[1] $\vdash P,\ P \Rightarrow Q$	by $\Rightarrow R$
$(P \Rightarrow Q) \Rightarrow P,\ \neg P,\ P \vdash Q$	by `notL`		
$(P \Rightarrow Q) \Rightarrow P,\ \neg P,\ P \vdash P$	by `axiom`	$P \vdash P,\ Q$	by `axiom`
[2.2] $\neg P,\ P \vdash P$	by `axiom`	[2] $P \vdash P$	by `axiom`

As we see, we get our alternative goal back after the `impR` rule, since we kept its negation in the hypotheses. So the moment 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 the `impL` in the second step, we wouldn't have been able to complete the proof anymore.

11.4 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. *Is it decidable?*
4. *Can a proof method get wedged?*
5. *What would we get if we were to drop the* `magic` *rule?*

We will investigate the first three questions later. It is very plausible that refinement logic is consistent – after all, we only weakened the rules and added a proven fact. Even the cut rule appears to be absolutely correct, but we need to check that. Adding `magic` and `cut` certainly makes it stronger and we have seen how we could prove a theorem in refinement logic with magic that we couldn't prove before. But before we prove that refinement logic is consistent and complete, let us look at the other two questions.

11.4.1 Can a proof method get wedged?

The example proof of Pierce's law seems to indicate that. Just imagine we had used a strategy that first tries to decompose the available formulas before attempting `magic`. This was a reasonable strategy for the tableau method and for Gentzen Systems. We could prove it to terminate. But when we apply it to Pierce's law, our proof gets stuck at the same goal where the calculus without `magic` got stuck.

```
      ⊢ ((P ⇒ Q) ⇒ Q) ⇒ 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
```

No application of the magic rule will help us here anymore.
The same holds for the two proof attempts for the law of excluded middle

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

Q: *So what is the reason? Why do proofs in refinement logic get stuck?*

The reason is that as soon as we apply the `impL` or the `orR`i rules, we have lost information that we had before.

Applying `impL` drops the conclusion that we had before and we're left with a completely different conclusion in the first subgoal. Formerly, we preserved the original goal as another alternative and we could decide later which one we wanted to prove. Now we make the choice as soon as we apply `impL`.

Similarly, the `orR`i rules force us to make choices early on in the proof and to drop the other alternative.

Both kinds of rules are *irreversible*, that is the subgoals are not equivalent to the main goal anymore but stronger. Applying the rule forces us to prove something stronger than the original goal and that may get us stuck.

11.4.2 What would we get if we were to drop the `magic` rule?

The original idea of the sequent calculus is to show that we can construct a proof for a goal G from a given set of hypotheses H. If G is a single conclusion, then the whole construction has to focus on that goal and the proof rules tell us how to construct *evidence* for the truth of G.

The `andR` rule, for instance tells us: "to prove $A \wedge B$ you have to prove both A and B" – evidence for A and evidence for B together is sufficient evidence for $A \wedge B$. The `orR`i rules for $A \vee B$ require us to make a choice between A and B. If we find evidence for either of them, we have sufficient evidence for $A \vee B$. `impR` is similar: if we show how to build evidence for B from evidence for A then we certainly have enough evidence for the implication $A \Rightarrow B$.

We also could view the goal G as a *task* that needs to be fulfilled and the rules tell us how to solve that task. "to solve $A \wedge B$ you must solve both A and B", etc.

A third view looks at G as an *event* that happens, assuming that the events in the list H have occurred.

All these views are very much influenced by a computer scientists view of logic. Formulas are not just truths, but tied to constructions. And it is the construction that interests us.

Refinement logic without the magic rule is highly constructive. It doesn't just tell us that something is true but gives us the precise reasons why. It shows us how to construct a solution. In fact, our Nuprl proof system, which uses this refinement logic, is capable of extracting this construction from a proof as executable algorithm.

Although refinement logic without `magic` is not complete with respect to propositional logic as defined in Smullyan's book (so-called *classical* logic), it represents a very well known logic, called *constructive*, *intuitionistic*, or *computational* logic, which is very relevant when reasoning about programs.

All this gets destroyed by the magic rule, which states that one of two alternatives, P or not P, must be true. We don't know which of the two, but we are assured that there is no third alternative, so one must be the case. Using this dogma makes it very easy to prove certain statements, since you don't have to worry how to provide the necessary evidence. But for a computer scientist, this dogma is very unsatisfactory.

Let me give you one example of a proof that uses this law in a very questionable, but absolutely correct way:

> *We want to prove that there are two irrational numbers x and y such that x^y is rational.*
>
> Consider $a = \sqrt{2}^{\sqrt{2}}$. Then a is either rational or not. In the first case choose $x = y = \sqrt{2}$, in the other choose $x = a$ and $y = \sqrt{2}$.
>
> That completes the proof but we still don't know what x and y are. We have been given two alternatives and told that one of the two is our solution.