Topic: Refinement Logic

Assignments Exercises:

Complete the missing cases of the proof

Extra: complexity of the simulation (watch for or)

Reading: Caldwell's Thesis for Tuesday
Smullyan Chapter IV, 43–52 for Thursday

8 Refinement Logic

On tuesday I have introduced you to Refinement Logic and the computer support for this logic offered by the NuPRL proof development system. In the next couple of weeks we will make sure that you can get access to NuPRL from the labs so that everybody has the chance to experiment with the system.
Refinement logic is the endpoint of a series of minor changes to the tableau calculus that we started with. Historically, this is not correct but if you present the calculi in the order

tableaux – blocked tableaux – multi-conclusioned sequent systems (or Gentzen Systems) – single-conclusioned sequent systems – Refinement Logic

then you can see that they are really related.
On Tuesday I pointed out the differences between tableaux and sequent systems and promised to prove that Refinement Logic can really prove the same theorems as Genzten systems. Before we do that, let me briefly summarize the differences between these calculi.

Gentzen Systems	Refinement Logic
multiple conclusions	single conclusion
set notation	list notation
—	cut
classical reasoning	constructive reasoning or axiom $A \vee \neg A$

I gave you a brief illustration why the axiom $A \vee \neg A$ – or the `magic` rule – is necessary. Today I want to prove that Single-Conclusioned Sequent Systems can do the same as Multi-Conclusioned ones, if you just add this `magic` rule.

Single-Conclusioned Sequent Systems + `magic` $\equiv$ Multi-Conclusioned Sequent System

I will proceed in two steps

- Multi-Conclusioned Sequent System $\equiv$ Single-Conclusioned Sequent Systems + `magic` + `cut`

  This is much easier to prove

- Single-Conclusioned + `cut` (+ `magic`) $\equiv$ Single-Conclusioned Sequent Systems (+ `magic`)

I will begin with the second, then we take a break for the quiz, and then I will show you how to prove the first.

8.1 Cut elimination theorem

The cut rule

```
H ⊢ G        cut A
  H ⊢ A
  H, A ⊢ G
```

is a very useful rule for structuring a proof. You introduce a fact A, or a lemma that you will use in the rest of the proof, and prove this fact separately. This is particularly useful if you use this lemma several times but also to keep the proof comprehensible by cutting it down into smaller pieces.
It is easy to see that this rule is correct but its disadvantage is, that it cannot be used in automated proofs. The reason is simple: all other rules depend on the outer structure of a formula on the left or right of the turnstyle. The cut rule can be applied ANYTIME and with ANY formula. Nobody has found a way of automating this properly – finding lemmata automatically is still a very difficult task.
So the question was – do we lose anything if we drop this rule? Can we prove more formulae if we use cuts? Well, the answer is: the cut rule is not really necessary – you can prove the same theorems without it.

Theorem 8.1 (Cut elimination)
Any formula provable in the single-conclusioned sequent calculus with cut has a single-conclusioned sequent proof that does not use the cut rule.

While this is easy to believe, a precise proof of this theorem is quite tricky. If you're interested you can look at Smullyan page 110–115, but I can't really recommend this because it is VERY dense.
So – why should we believe we can do without the cut rule? Essentially, it is because *we have to prove the formula that we use*, so we can *use the full proof of A, whenever A is used.* Sounds plausible, doesn't it?
It is – in fact it is the *central idea* of the proof of the cut elimination theorem. But a precise proof that uses this idea is difficult to formulate and you have to look at many details. Let me illustrate why.
Assume we have used the cut rule

```
H ⊢ G      cut A
  H ⊢ A
  H, A ⊢ G
```

to prove a goal G and let pf be the sequent proof for the fact A. Now let us look at the proof for $H, A \vdash G$. If A would remain untouched during that proof and we would just use it at the end in a `hypothesis` rule $H, A, H' \vdash A$ then we could replace this rule application by the proof pf.
But imagine if A has the form $p \wedge q$ and A is decomposed somewhere in the proof before being used. Then we would have to deal with rules of the form $H, p, H' \vdash p$. Now we have to analyze the proof pf for A. If it used the rule `andR` to decompose A into p and q, we could simply take the proof up to that point, drop the `andR` rule and the proof part for q and insert the rest as proof for p instead of the `hypothesis` rule. If we didn't use `andR`, then H must have contained a superformula of A that was decomposed into an occurrence of A on the left hand side. In this case we could further decompose this A into p and q and insert this whole proof before applying the `hypothesis` rule.

Now you have to think about the other connectives: disjunction, implication, negation and do the same. And to be precise, you have to make an inductive proof because A may be a nested formula that is decomposed into very small pieces.
This tells you how to eliinate a single cut rule from a proof. Now you proceed inductively and eliminate all the cuts from bottom to top – assuming that after elimination the remaining proof branch is cut-free.
Of course, afterwards you think about how to make this proof more elegant. So this is not an easy proof but you get a rough idea how to proceed.
There is a cost to eliminating cuts. Our proofs will grow. In fact they can grow more that super-exponentially – in some cases the size reduction that you can achive by introducing cuts is non-elementary. This of course holds only for pathological cases. But nevertheless you should know.

8.2 Quiz

Which of the following two matrices represents a tautology?

1. The first
2. The second
3. Both
4. None

$$\begin{bmatrix} \bar{p} & p \\ \bar{q} & q \end{bmatrix} \qquad \begin{bmatrix} [\bar{p} & p] \\ [\bar{q} & q] \end{bmatrix}$$

8.3 Consistency

Now let us look at the properties of Refinement Logic. First, let me just state

Theorem 8.2
The Refinement Logic calculus is consistent

The reason is that we can simulate every RL-rule by a multi-succedent rule. Just look at your handouts:
All the right rules except for `orR1` and `orR2` are the same, if you had only one conclusion to start with. The `andL`, `orL`, and `hypothesis` rule are the same as well. `orR1`, `orR2`, `impL`, and `notL` are subsumed by the multi-succedent rules, because these rules are the same except that they keep more than one conclusion around. So if you can prove this goal without using Γ then you can surely prove it with Γ being around – because you don't have to use Γ.
Because of the cut elmination theorem we don't need the cut rule in a proof, which means that this rule can be replaced by all the others. The `lemma` rule just refers to a proof that we did externally – we could simply insert it here. So what remains is the `magic` rule, which we simulate as follows:

```
⊢ A ∨ ¬A          by orR
1. ⊢ A, ¬A          by notR
1.1. A ⊢ A            by axiom
```

What about `falseL`? Its a bit tricky because it is more basic. Essentially `false` is contradiction – like `A ∧ ¬A`. We'll use that in the proof
– *Do this as exercise??*

```
false ⊢ G                                  by ClassCases A
1. false, A ⊢ G                            by cut ¬A
1.1. false, A ⊢ ¬A                           by notR
1.1.1. false, A, A ⊢ false                  by hypothesis 1
1.2. false, A, ¬A ⊢ G                        by notL 3
1.2.1 false, A, ¬A ⊢ A                       by hypothesis 2
2. false, ¬A⊢ G                            by cut ¬¬A
2.1. false, ¬A ⊢ ¬¬A                         by notR
2.1.1. false, ¬A, ¬A ⊢ false                by hypothesis 1
2.2. false, ¬A, ¬¬A ⊢ G                      by andL 3
2.2.1 false, ¬A, ¬¬A ⊢ ¬A                     by hypothesis 2
```

So we can simulate all the RL rules in the multi-conclusioned calculus.
Exercise: Give a precise (inductive) proof for the consistency of RL

8.4 Completeness

Now let us show completeness. Can we really do everything in refinement logic that we could do in multi-conclusioned sequent systems?

Theorem 8.3 *Any formula provable by tableau can be proved in the single-conclusioned sequent calculus with magic and cut.*

This seems plausible, but how do we prove it? Assuming that we start with a single conclusion – *can we simulate the multi-conclusioned rules by single-conclusioned ones?*
Look at the handout and compare the rules: There is no te every RL-rule by a multi-succedent rule. Just look at your handouts: `andL`, `andR`, `orL`, `impR`, `notR`, and `axiom` are really the same. But the other three rules *create* an additional conclusion, so we can't simulate them directly.

So we have to do something different. If we cannot express the rules directly, we have to *mimic a multi-conclusioned proof* with just one conclusion. Look at the meaning of multiple conclusions. We have said that $\Delta \vdash \Gamma$ should be understood as: "under the assumption that all the formulae in Δ are true we can show that one of the formulae in Γ is true". So if Γ is $G_1, ..., G_n$ then

proving $\Delta \vdash \Gamma$ is the same as proving $\Delta \vdash G_1 \vee G_2 ... \vee G_n$

We will use this idea to simulate multi-conclusioned proofs in Refinement logic. That is, we will express every multi-succedent rule that operates on $\Delta \vdash G_1, G_2 ..., G_n$ by a series of refinement logic rules that operate on $\Delta \vdash G_1 \vee G_2 ... \vee G_n$. Let me introduce an abbreviation:

For $\Gamma = G_1, G_2 ..., G_n$ we define Γ^* to be the formula $G_1 \vee (G_2 \vee (... \vee G_n))$.

I put the parentheses around the disjunctions to avoid ambiguities. If I omit parentheses, I mean exactly this way of parenthesizing.
With this abbreviation we can state our simulation theorem

Theorem 8.4
Let $\Delta = H_1, ..., H_m$ *and* $\Gamma = G_1, G_2 ..., G_n$ *be lists of formulae. If* $\Delta \vdash \Gamma$ *is provable in the multi-conclusioned sequent calculus, then* $\Delta \vdash G_1 \vee (G_2 \vee (... \vee G_n))$ *is provable in refinement logic. In particular, if* $\Delta \vdash G_1$ *is provable in the multi-conclusioned sequent calculus then it is also provable in refinement logic.*

Note that I said "lists of formulae" and not "sets". Although multi-succedent rules operate on sets, we can always assume that we have a standard way of listing the elements of these sets so that we know how Γ^* is related to $\Gamma = G_1, G_2..., G_n$. I need this only for proof purposes – I don't change the calculus.

Proof: We will proceed by induction over the depth of the multi-conclusioned sequent proof and show how to simulate each proof step in refinement logic.

Base Case: If the depth of the proof for $\Delta \vdash \Gamma$ is 1 then the proof must be the application of the `axiom` rule to some G_i which also occurs in Δ as H_j.

$H_1, .G_i = H_j.., H_m \vdash G_1, G_2, .., G_i, ., G_n$

We can essentialy do the same in refinement logic by using the rule `hypothesis` j.

But we must keep in mind that the multi-conclusioned sequent rules can operate on ANY element of Γ, while we need to prove

$H_1, .G_i = H_j.., H_m \vdash G_1 \vee (G_2 \vee (..(G_i \vee .. \vee G_n)))$

that is we have a goal formula with a fixed structure. So, to isolate G_i, we must move it to the left of the disjunction.

We need three derived proof rules for this purpose.

```
Δ ⊢ A∨(B∨C)     by orAssocL     Δ ⊢ (A∨B)∨C     by orAssocR
  Δ ⊢ (A∨B)∨C                     Δ ⊢ A∨(B∨C)

Δ ⊢ A∨B         by orCommut     Δ ⊢ (A∨B)∨C     by orCommutL
  Δ ⊢ B∨A                         Δ ⊢ (B∨A)∨C
```

These rules can be derived in Refinement Logic. For instance:

```
   Δ ⊢ A∨B                   by cut B∨A
1.    Δ ⊢ B∨A
2.    Δ, B∨A ⊢ A∨B            by orL m+1
2.1.   Δ, A ⊢ A∨B            by orR1
2.1.1.   Δ, A ⊢ A            by hypothesis m+1
2.2.   Δ, B ⊢ A∨B            by orR2
2.2.1.   Δ, B ⊢ B            by hypothesis m+1
```

So we can write a tactic that performs all these steps and we can consider `orCommut` as a derived inference rule.

Exercise: Prove that the rule `orAssocL` can be derived in Refinement Logic

By applying the three rules we can transform

$H_1, .G_i = H_j.., H_m \vdash G_1 \vee (G_2 \vee (..(G_i \vee .. \vee G_n)))$

into

$H_1, .G_i = H_j.., H_m \vdash G_i \vee (G_1 \vee (G_2 \vee (..(G_{i-1} \vee (Gi{+}1 \vee .. \vee G_n)))))$

and then proceed by applying `orR1` to get

$H_1, .G_i = H_j.., H_m \vdash G_i$

and now we can apply `hypothesis` j.

Let me show you how this works when you want to isolate G_2 out of $G_1 \vee (G_2 \vee G_3)$.

```
     Δ ⊢ G1∨(G2∨G3)      by orAssocL
1.   Δ ⊢ (G1∨G2)∨G3      by orCommutL
1.1. Δ ⊢ G2∨(G1∨G3)      by orAssocR
```

Exercise: For $\Gamma = G_1, G_2..., G_n$ define Γ_i^* to be the formula $G_1 \vee (G_2 \vee (...(G_{i-1} \vee (Gi{+}1 \vee ...G_n))))$
Describe a tactic that "implements" the derived rules

$$\frac{\Delta \vdash \Gamma^*}{\Delta \vdash G_i \vee \Gamma_i^*}\ \texttt{SwapOut}\ i \qquad \frac{\Delta \vdash G_i \vee \Gamma_i^*}{\Delta \vdash \Gamma^*}\ \texttt{SwapIn}\ i$$

Try to use the `THEN, THENL, ORELSE, Repeat` notation where possible

Induction case: Assume that for all Δ,Γ, and all multi-conclusioned sequent proofs for $\Delta \vdash \Gamma$ of depth n we find a refinement logic proof for $\Delta \vdash \Gamma^*$.

We show how to convert a multi-conclusioned sequent proof of depth $n{+}1$ into a refinement logic proof by looking at the FIRST proof rule that was applied.

`andL`: The first step in the deduction was

$H_1, .H_j = \mathtt{A} \wedge \mathtt{B}.., H_m \vdash \Gamma$ by `andL`
  $H_1, .H_j = \mathtt{A,B}.., H_m \vdash \Gamma$

We can simulate this first step by

$H_1, .H_j = \mathtt{A} \wedge \mathtt{B}.., H_m \vdash \Gamma^*$ by `andL` j
  $H_1, .H_j = \mathtt{A,B}.., H_m \vdash \Gamma^*$

By the induction assumption we can find a refinement proof for this subgoal because the multi-conclusioned sequent proof for $H_1, .H_j = \mathtt{A,B}.., H_m \vdash \Gamma$ has depth n.

So this was easy.

`andR`: The first step in the deduction was

$\Delta \vdash G_1, ..G_{i-1},\ A \wedge B,\ Gi{+}1, ..G_n$ by `andR`
  $\Delta \vdash G_1, ..G_{i-1},\ A,\ Gi{+}1, ..G_n$
  $\Delta \vdash G_1, ..G_{i-1},\ B,\ Gi{+}1, ..G_n$

We need to show that we can express the following rule in refinement logic:

$\Delta \vdash G_1 \vee ..G_{i-1} \vee\ A \wedge B\ \vee\ Gi{+}1 \vee\ .. \vee G_n$
  $\Delta \vdash G_1 \vee ..G_{i-1} \vee\ A\ \vee\ Gi{+}1 \vee\ .. \vee G_n$
  $\Delta \vdash G_1 \vee ..G_{i-1} \vee\ B\ \vee\ Gi{+}1 \vee\ .. \vee G_n$

By the induction assumption we can find a refinement prof for the "subgoals" because the corresponding multi-succedent proofs have depth n.

It is sufficient to describe the following derived rule:

$\Delta \vdash A \wedge B\ \vee\ \Gamma_i^*$ `andR*`
  $\Delta \vdash A\ \vee\ \Gamma_i^*$
  $\Delta \vdash B\ \vee\ \Gamma_i^*$

because if we apply `SwapOut` i before this rule and `SwapIn` i afterwards, we have exactly the rule we want to have. So in the rest of this proof we won't bother about this distinction anymore but always assume that we operate on the first formula of Γ^*.

So how do we implement `andR`*? We first cut in the desired subgoals $A\ \vee\ \Gamma_i^*$ and $B\ \vee\ \Gamma_i^*$ and then show that they are sufficient to prove $A \wedge B\ \vee\ \Gamma_i^*$:

$\Delta \vdash A\wedge B \ \vee\ \Gamma_i^*$ by cut $(A \vee \Gamma_i^*) \wedge (B \vee \Gamma_i^*)$

1\. $\Delta \vdash (A \vee \Gamma_i^*) \wedge (B \vee \Gamma_i^*)$ by andR

1.1. $\Delta \vdash A \vee \Gamma_i^*$

1.2. $\Delta \vdash B \vee \Gamma_i^*$

2\. $\Delta,\ (A \vee \Gamma_i^*) \wedge (B \vee \Gamma_i^*) \vdash A\wedge B \vee \Gamma_i^*$ by andL

2.1. $\Delta,\ A \vee \Gamma_i^*,\ B \vee \Gamma_i^* \vdash A\wedge B \vee \Gamma_i^*$ by orL 1

2.1.1. $\Delta,\ A,\ B \vee \Gamma_i^* \vdash A\wedge B \vee \Gamma_i^*$ by orL 2

2.1.1.1. $\Delta,\ A,\ B \vdash A\wedge B \vee \Gamma_i^*$ by orR1

2.1.1.1.1. $\Delta,\ A,\ B \vdash A\wedge B$ by andR

2.1.1.1.1.1. $\Delta,\ A,\ B \vdash A$ by hypothesis m+1

2.1.1.1.1.2. $\Delta,\ A,\ B \vdash B$ by hypothesis m+2

2.1.1.2. $\Delta,\ A,\ \Gamma_i^* \vdash A\wedge B \vee \Gamma_i^*$ by orR2

2.1.1.2.1. $\Delta,\ A,\ \Gamma_i^* \vdash \Gamma_i^*$ by hypothesis m+2

2.1.2. $\Delta,\ \Gamma_i^*,\ B \vee \Gamma_i^* \vdash A\wedge B \vee \Gamma_i^*$ by orR2

2.1.2.1 $\Delta,\ \Gamma_i^*,\ B \vee \Gamma_i^* \vdash \Gamma_i^*$ by hypothesis m+1

So we have found an "implementation" of andR*

`orL:` Direct simulation as in the case of andL

`orR:` Assuming that we operate on the first formula of the set the first step was

$\Delta \vdash A\vee B,\ \Gamma$ by orR
$\quad \Delta \vdash A,\ B,\ \Gamma$

We need to show that we can express the following rule in refinement logic:

$\Delta \vdash A\vee B \ \vee\ \Gamma^*$ orR*
$\quad \Delta \vdash A\vee(B \vee \Gamma^*)$

So orR* is the same as orAssocR

`impL:` The first step in the deduction has been

$H_1,.H_j = \mathtt{A}\Rightarrow\mathtt{B}..,H_m \vdash \Gamma$ by impL
$\quad H_1,.H_j = \mathtt{A}\Rightarrow\mathtt{B}..,H_m \vdash A,\Gamma$
$\quad H_1,.H_j = \mathtt{B}..,H_m \vdash \Gamma$

We can simulate this first step by

$H_1,.H_j = \mathtt{A}\Rightarrow\mathtt{B}..,H_m \vdash \Gamma$ by impL*
$\quad H_1,.H_j = \mathtt{A}\Rightarrow\mathtt{B}..,H_m \vdash A\vee\Gamma$
$\quad H_1,.H_j = \mathtt{B}..,H_m \vdash \Gamma$

To implement impL* we proceed as before

$H_1,.H_j = \mathtt{A}\Rightarrow\mathtt{B}..,H_m \vdash \Gamma$ by cut $(A\vee\Gamma) \wedge (B\Rightarrow\Gamma)$

1\. $H_1,.H_j = \mathtt{A}\Rightarrow\mathtt{B}..,H_m \vdash (A\vee\Gamma) \wedge (B\Rightarrow\Gamma)$ by andR

1.1. $H_1,.H_j = \mathtt{A}\Rightarrow\mathtt{B}..,H_m \vdash A\vee\Gamma$

1.2. $H_1,.H_j = \mathtt{A}\Rightarrow\mathtt{B}..,H_m \vdash B\Rightarrow\Gamma$ by impR

1.2.1 $H_1,.H_j = \mathtt{A}\Rightarrow\mathtt{B}..,H_m, B \vdash \Gamma$

2\. $H_1,.H_j = \mathtt{A}\Rightarrow\mathtt{B}..,H_m,\ (A\vee\Gamma) \wedge (B\Rightarrow\Gamma) \vdash \Gamma$ by andL m+1

2.1. $H_1,.H_j = \mathtt{A}\Rightarrow\mathtt{B}..,H_m,\ A\vee\Gamma,\ \ B\Rightarrow\Gamma \ \vdash \Gamma$ by orL m+1

2.1.1. $H_1,.H_j = \mathtt{A}\Rightarrow\mathtt{B}..,H_m,\ A,\ \ B\Rightarrow\Gamma \ \vdash \Gamma$ by impL j

2.1.1.1. $H_1,.H_j = \mathtt{A}\Rightarrow\mathtt{B}..,H_m,\ A,\ \ B\Rightarrow\Gamma \ \vdash A$ by impL $j$hypothesis m+1

2.1.1.2. $H_1,.H_j = \mathtt{B}..,H_m,\ A,\ \ B\Rightarrow\Gamma \ \vdash \Gamma$ by impL m+2

2.1.1.2.1. $H_1,.H_j = \mathtt{B}..,H_m,\ A,\ \ B\Rightarrow\Gamma \ \vdash B$ by hypothesis j

2.1.1.2.2. $H_1,.H_j = \mathtt{B}..,H_m,\ A,\ \ \Gamma \ \vdash \Gamma$ by hypothesis j

2.1.2. $H_1,.H_j = \mathtt{A}\Rightarrow\mathtt{B}..,H_m,\ \Gamma,\ \ B\Rightarrow\Gamma \ \vdash \Gamma$ by hypothesis m+1

`impR:` The first step in the deduction has been

$\Delta \vdash A\Rightarrow B,\ \Gamma$ by impR
$\quad \Delta,\ A \vdash B,\ \Gamma$

We need to show that we can express the following rule in refinement logic:

$\Delta \vdash A\Rightarrow B \ \vee\ \Gamma^*$ impR*
$\quad \Delta,\ A \vdash B \vee \Gamma^*$

This one I leave as exercise

$\Delta \vdash A\Rightarrow B \ \vee\ \Gamma^*$ by cut $A \Rightarrow (B \vee \Gamma^*)$

1\. $\Delta \vdash A \Rightarrow (B \vee \Gamma^*)$ by impR

1.1. $\Delta,\ A \vdash B \vee \Gamma^*$

2\. $\Delta,\ A \Rightarrow (B \vee \Gamma^*) \vdash A\Rightarrow B \vee \Gamma^*$ by ClassCases A

2.1. $\Delta,\ A \Rightarrow (B \vee \Gamma^*),\ A \vdash A\Rightarrow B \vee \Gamma^*$ by impL m+1

2.1.1. $\Delta,\ A \Rightarrow (B \vee \Gamma^*),\ A \vdash A$ by hypothesis m+2

2.1.2. $\Delta,\ B \vee \Gamma^*,\ A \vdash A\Rightarrow B \vee \Gamma^*$ by orL m+1

2.1.2.1. $\Delta,\ B,\ A \vdash A\Rightarrow B \vee \Gamma^*$ by orR1

2.1.2.1.1 $\Delta,\ B,\ A \vdash A\Rightarrow B$ by impR

2.1.2.1.1 $\Delta,\ B,\ A,\ A \vdash B$ by hypothesis m+1

2.1.2.2. $\Delta,\ \Gamma^*,\ A \vdash A\Rightarrow B \vee \Gamma^*$ by orR2

2.1.2.2.1. $\Delta,\ \Gamma^*,\ A \vdash \Gamma^*$ by hypothesis m+2

2.2. $\Delta,\ A \Rightarrow (B \vee \Gamma^*),\ \neg A \vdash A\Rightarrow B \vee \Gamma^*$ by orR1

2.2.1. $\Delta,\ A \Rightarrow (B \vee \Gamma^*),\ \neg A \vdash A\Rightarrow B$ by impR

2.2.1.1. $\Delta,\ A \Rightarrow (B \vee \Gamma^*),\ \neg A,\ A \vdash B$ by notL m+2

2.2.1.1.1. $\Delta,\ A \Rightarrow (B \vee \Gamma^*),\ \neg A,\ A \vdash A$ by hypothesis m+3

`notL:` The first step in the deduction has been

$H_1,.H_j = \neg\mathtt{A}..,H_m \vdash \Gamma$ by notL
$\quad H_1,.H_j = \neg\mathtt{A}..,H_m \vdash A,\Gamma$

We can simulate this first step by

$H_1,.H_j = \neg\mathtt{A}..,H_m \vdash \Gamma$ by notL*
$\quad H_1,.H_j = \neg\mathtt{A}..,H_m \vdash A\vee\Gamma$

Since $\neg A$ can be viewed as abbreviation for $A \Rightarrow$ false this works precisely as impL*

`notR:` The first step in the deduction has been

$\Delta \vdash \neg A,\ \Gamma$ by notR
$\quad \Delta,\ A \vdash \ \Gamma$

We need to show that we can express the following rule in refinement logic:

$\Delta \vdash \neg A \ \vee\ \Gamma^*$ notR*
$\quad \Delta,\ A \vdash \Gamma^*$

Since $\neg A$ can be viewed as abbreviation for $A \Rightarrow$ false this works precisely as impR*

$\Delta \vdash \neg A \ \vee\ \Gamma^*$ by cut $A \Rightarrow \Gamma^*$

1\. $\Delta \vdash A \Rightarrow \Gamma^*$ by impR

1.1. $\Delta,\ A \vdash \Gamma^*$

2\. $\Delta,\ A \Rightarrow \Gamma^* \vdash \neg A \vee \Gamma^*$ by ClassCases A

2.1. $\Delta,\ A \Rightarrow \Gamma^*,\ A \vdash \neg A \vee \Gamma^*$ by impL m+1

2.1.1. $\Delta,\ A \Rightarrow \Gamma^*,\ A \vdash A$ by hypothesis m+2

2.1.2. $\Delta,\ \Gamma^*,\ A \vdash \neg A \vee \Gamma^*$ by orR2

2.1.2.1. $\Delta,\ \Gamma^*,\ A \vdash \Gamma^*$ by hypothesis m+1

2.2. $\Delta,\ A \Rightarrow \Gamma^*,\ \neg A \vdash \neg A \vee \Gamma^*$ by orR1

2.2.1. $\Delta,\ A \Rightarrow \Gamma^*,\ \neg A \vdash \neg A$ by hypothesis m+3

Exercise: show how to implement the rule `impR*`.
Hint: use the `magic` or `ClassCases` rule.

So in all cases we have shown how to simulate multi-succedent proofs using refinement logic. We need the `magic` rule only for implication and negation on the right, so we see where it is actually needed to achieve completeness. So as a result we get:

Theorem 8.5 (Completeness of Refinement Logic)
The Refinement Logic calculus is complete

I have presented the inductive proof in a rule-fashion because this enables us to write an algorithm that converts multi-succedent proofs into refinement logic proofs. Because of the rules `SwapOut` and `SwapIn` this proof will in the end be much bigger than the original multi-succedent proof. It gets even worse, if we try to eliminate the cuts as well.
But there is no way out. Although refinement logic proofs are sufficient to prove that a formula is a tautology, they cannot be as short as multi-succedent proofs. In the worst case, they must be exponentially longer. For those of you who are interested, here is an example:
For $n \geq 0$ consider the propositional formula class F_n

$$F_n \quad \equiv \quad A_n \wedge \bigwedge_{i=1}^{n-1} \underbrace{(B_{i-1} \Rightarrow (B_i \vee A_i) \vee B_i)}_{O_i} \wedge \underbrace{((B_0 \vee A_0) \vee B_0)}_{O_0}$$

$$\Rightarrow \quad A_0 \vee \bigvee_{i=0}^{n-1} \underbrace{B_i \wedge A_{i+1}}_{N_i}$$

Now if you look at the properties of these formulae you find out

- F_0 needs one leaf in $\mathcal{LJ}_{mc}$ and $\mathcal{LJ}$
- there exist an $\mathcal{LJ}_{mc}$ proof of F_n with $6n - 2$ leaves (branches)
- F_n implies a reduction ordering $O_j \longmapsto O_{j+1} \longmapsto N_j$ on $\mathcal{LJ}$ proofs
- each $\mathcal{LJ}$ proof of F_n requires at least $5(2^n - 1)$ leaves

This is a fairly new result but it describes a pathological case. In the average the translation gets much better results.