Restart Rule

The *restart rule* states that at any point during a goal-directed proof we may replace the current goal with any of the previous goals.

The set of previous goals is called the *history*.

Motivating the Restart Rule

1. Suppose we want to show $\mathcal{P} \vdash A$ and get stuck at $\mathcal{P}^* \vdash B$ after some proof steps.

2. Because for any A either A or $\neg A$ holds, we *could have* started with two proofs $\mathcal{P}, A \vdash A$ and $\mathcal{P}, \neg A \vdash A$ instead. The former directly succeeds. Hence, we *could have* started with $\mathcal{P}, \neg A \vdash A$.

3. If we start with $\mathcal{P}, \neg A \vdash A$, but never touch $\neg A$, we will eventually get stuck at $\mathcal{P}^*, \neg A \vdash B$ – just as in (1).

4. If we manage to show $\mathcal{P}^*, \neg A \vdash A$ (*restart rule!*) we know that both A and $\neg A$ (trivial) follow from $\mathcal{P}^*, \neg A$. Hence, also $\mathcal{P}^*, \neg A \vdash B$ holds.

Summary of Rules

1. If the current (atomic) goal can be found in the data, **succeed**.
2. If $\bot$ can be found in the data, **succeed**.
3. **Implication:** to show $\mathcal{P} \vdash A \Rightarrow B$ show $\mathcal{P}, A \vdash B$.
4. **Conjunction:** to show $\mathcal{P} \vdash A \wedge B$ show $\mathcal{P} \vdash A$ and $\mathcal{P} \vdash B$.
5. **Atoms:** to show $\mathcal{P}, A \Rightarrow B \vdash B$ (B atomic) show $\mathcal{P}, A \Rightarrow B \vdash A$.
6. **Atoms (with $\bot$):** to show $\mathcal{P}, A \Rightarrow \bot \vdash B$ (B atomic) show $\mathcal{P}, A \Rightarrow \bot \vdash A$.
7. **Restart:** replace the current goal with a previous one.

Remark. All formulas need to be ready for computation (the goal could be a conjunction of such formulas).

Soundness and Completeness

Strictly speaking, before using a proof system **X** we we should prove that it is *sound* and *complete*.

- **Soundness:** if **X** says that argument $\mathcal{P} \vdash A$ is valid then it really is (i.e. $\bigwedge \mathcal{P} \Rightarrow A$ is a tautology according to truth tables)
- **Completeness:** if an argument is valid (again, according to truth tables) then **X** can deliver a proof for it

Proving completeness is usually very difficult (textbook pp. 133–142).

Proving soundness means showing for each proof rule that if it is applied to a valid argument then the result will also be a valid argument. This is quite similar to the informal explanations given when the rules have been introduced. Try to get the gist of the proof on pp. 132–133 in the textbook.

Mechanical Reasoning

For a proof system to be implementable, soundness and completeness are not enough. We should also make sure that proofs always *terminate* at some point and return with a result (either "yes, valid" or "no, not valid").

For example, we shouldn't apply the restart rule again and again with the same goal if that doesn't get us anywhere.

One way of achieving termination for the goal-directed system would be to combine it with a *loop checker*.

Refinements

To guarantee termination (without the necessity of loop checking) we could also make the following refinements:

1. Only restart if nothing else is possible.
2. Restart only with atoms.
3. Only restart B, if directly after that we can apply a rule different from restart. Hence, check either B or $A \Rightarrow B$ (for some A) is in the data. In the former case we would succeed in the next step, in the latter case we could apply the rule for atoms in the next step. There are no other cases.
4. When applying one of the rules for atoms, the implication we used can be deleted from the data.

Efficiency

Ideally a proof procedure should not only terminate, but it should terminate as quickly as possible.

A lot of effort in research on *automated reasoning* is devoted towards making theorem provers more *efficient*. Work is done on all levels:

- on the logical level,
- on the algorithmic level, and
- on the implementation level.

Exercise

1. Translate the following formula into a conjunction of formulas that are ready for computation:

$$(P \Rightarrow (Q \vee R)) \wedge \neg Q$$

2. Now think about how you could get a computer to do the same thing.

Systematic Translation

$$
\begin{aligned}
(A\Rightarrow(B \wedge C))^* &= (A\Rightarrow B)^* \wedge (A\Rightarrow C)^* \\
(A\Rightarrow(B\Rightarrow C))^* &= (A \wedge B\Rightarrow C)^* \\
(A\Rightarrow(B \vee C))^* &= ((A \wedge (B\Rightarrow\bot))\Rightarrow C)^* \\
(A\Rightarrow\neg B)^* &= ((A \wedge B)\Rightarrow\bot)^* \\
(A\Rightarrow q)^* &= A^*\Rightarrow q, \quad q \text{ atomic} \\
(A\Rightarrow\bot)^* &= A^*\Rightarrow\bot \\
(\neg A)^* &= (A\Rightarrow\bot)^* \\
(A \vee B)^* &= ((A\Rightarrow\bot)\Rightarrow B)^* \\
(A \wedge B)^* &= A^* \wedge B^*
\end{aligned}
$$

Summary: Goal-directed Deduction

- Formulas need to be *ready for computation* (for humans the four simple translation rules are enough).
- There are *seven goal-directed rules*. Remember them!
- If carefully formulated, the goal-directed system is suited for implementation purposes.

Still to come:

- Implementing a goal-directed theorem prover in Prolog
- Goal-directed deduction for first-order predicate logic
- Correspondence of goal-directed proofs and Prolog's query evaluation process (if time permits)