Example 7.4: With the notation as in Example 7.2, define a system $\langle U, N \rangle$ by setting $N = \{r_1, r_3\}$. If $S = \{\beta\}$, then $C_S(\emptyset) = \{\alpha\}$ is not deductively closed as it does not contain γ in violation of rule r_3.

Proposition 7.5: *If $S \subseteq C_S(I)$, then $C_S(I)$ is deductively closed.*

Proof: Suppose all the premises of a rule r with conclusion φ are in $C_S(I)$ and all of r's restraints are outside it. By the definition of $C_S(I)$, we can produce an S–deduction containing all the premises of r. All of the restraints in r are outside S by hypothesis. We can thus extend the S–deduction to one of φ by applying r to get $\varphi \in C_S(I)$ as desired. □

Definition 7.6: $S \subseteq U$ is an *extension of* I if $C_S(I) = S$. S is an *extension* if it is an extension of the empty set $\emptyset$.

The extensions S of I are the analogs for nonmonotonic systems of the logical consequences of I. Every member of an extension is deducible from I and all the S–consequences of I are in fact in S. We give some basic properties of extensions in Exercises 3–5.

It turns out that extensions capture many procedures in mathematics and computer science. We give some mathematical examples in Exercises 8–9. Now we return to PROLOG programs with negation and their connection to extensions through the notion of stable models.

From our current point of view, it is natural to try to consider the negation as failure as a nonmonotonic system. The underlying idea of negation as failure as presented in the last section is that we may assert $\neg p$ when we do not know (cannot deduce) p. This suggests a natural translation of a general PROLOG program into a nonmonotonic formal system.

Recall from Definition 6.11 that a general program clause has the form $p :\!-\ q_1, \ldots, q_n, \neg s_1, \ldots, \neg s_m$ where p, q_i and s_j are atoms.

Remember also that we are in the propositional case. Thus, if a program of interest has variables, we consider instead all ground instances of the program clauses in the Herbrand universe. We can now easily translate a general program P containing only ground atoms into a nonmonotonic formal system in a natural way. We consider each ground atom as a propositional letter. These atoms constitute our universe U. Each program clause C of P of the form $p :\!-\ q_1, \ldots, q_n, \neg s_1, \ldots, \neg s_m$ is translated into a rule $tr(C)$:

$$\frac{q_1, \ldots, q_n : s_1, \ldots, s_m}{p}.$$

The nonmonotonic system is then specified by letting its set of rules N be the collection $\{tr(C) : C \in P\}$ of translations of clauses of P.

Definition 7.7: Let P be a general program with only ground clauses. $tr(P)$, the *translation of* P, is the nonmonotonic system $\langle U, N \rangle$ where U is the set of atoms appearing in P and $N = \{tr(C) : C \in P\}$ is the set of translations of clauses of P.

It turns out that it is precisely the extensions of $tr(P)$ which are the stable models of P introduced by Gelfond and Lifschitz [1988, 5.4] to capture a strong notion of semantics for general programs with negation as failure.

Definition 7.8: If U is the set of atoms appearing in a ground general program P and $M \subseteq U$, then P_M is the program obtained from P by deleting each clause that has in its body a negative literal $\neg s$ with $s \in M$ and also deleting all negative literals in the bodies of the remaining clauses. As P_M is clearly a PROLOG program (it has no negative literals), it has a unique minimal Herbrand model by Exercise II.10.3. A *stable model* of P is an $M \subseteq U$ such that M is the unique minimal Hebrand model of P_M.

This terminology is justified by the following theorem which shows that stable models of P are in fact models of P.

Theorem 7.9: *Every stable model of P is a minimal model of P.*

Proof: Suppose M is a stable model of P. Consider a clause C of P of the form $p :\!-\ q_1, \ldots, q_n, \neg s_1, \ldots, \neg s_m$. If some $s_j \in M$, then M trivially satisfies C. If none of the s_j are in M, then $p :\!-\ q_1, \ldots, q_n$ is in P_M. As M is a model of P_M, $p \in M$ if $q_1, \ldots, q_n \in M$. Thus M satisfies C in this case as well and so M is a model of P.

To see that M is a minimal model of P, consider any $M' \subseteq M$ which is also a model of P. We need to show that $M = M'$. By the definition of a stable model, it suffices to show that M' is a model of P_M. Now any clause C' of P_M comes from some C in P as above with $s_j \notin M$ for $1 \leq j \leq m$. It is then of the form $p :\!-\ q_1, \ldots, q_n$. Suppose then that $q_1, \ldots, q_n \in M'$. We need to show that $p \in M'$. As $M' \subseteq M$, $s_j \notin M'$ for every $1 \leq j \leq m$. Thus, as M' is a model of $C \in P$, $p \in M'$ as required. □

Example 7.10: Let Q be the following general program of Gelfond and Lifschitz [1988, 5.4]:

$$p(1,2).$$
$$q(x) :\!-\ p(x,y), \neg q(y).$$

Q has two minimal Herbrand models: $M_1 = \{p(1,2), q(1)\}$ and $M_2 = \{p(1,2), q(2)\}$ (Exercise 6). The usual negation as failure rule applied to this program answers "no" to the question "?- $q(2)$." but "yes" to the question "?- $q(1)$." Thus we should prefer the first model over the second.

Now consider the possible subsets of the Herbrand universe as candidates for stable models of the ground instances of this program. First, the program itself is transformed into the following ground version P: