Guidelines for Understanding Negation as Failure

Negation As Failure (\+ in Prolog), abbreviated as NAF, expresses logical negation only for *ground* goals (i.e., goals where all unknowns are solved in terms of literals). Therefore,

Guideline 1: where the correctness of a program *requires* logical negation, use "safe" negation -- i.e., NAF in conjunction with testing the goal to be ground (a definition of this predicate appears in our class directory), and to this end place such goals at the end of goal lists so variables will be solved before they are reached.

Though they do not express logical negation, NAF goals can be very useful when the goal contains unsolved variables. Their value lies in understanding such goals as providing us with the ability to express a different form of logical condition in our queries. Hence,

Guideline 2: where NAF is used in the presence of unsolved variables, understand the success of \+Goal to mean "*for all* unbound variable values, Goal fails"; that is, a *universally quantified* negative goal; such goals don't solve for variable values, rather their success confirms that none exist (their failure indicates that values do exist, but those values are not revealed).

From the logic perspective, a goal \+Goal is treated by pausing to solve Goal, then reversing the outcome. Hence the logical effect is solve $\exists X.Goal$ and then return $\neg(\exists X.Goal)$ which is logically equivalent to $\forall X.(\neg Goal)$.