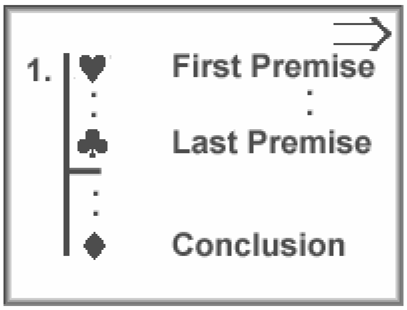
CHAPTER 7B
PROOFS: OTHER

THE SPECIAL RULES

PREMISES AND CONCLUSION

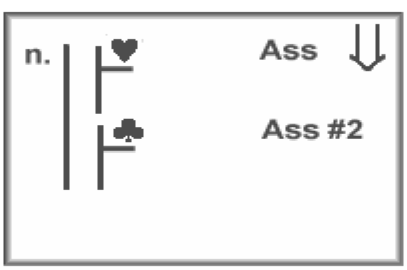


The rules Premise, Assumption, and Ass#2, let you make assumptions. Every problem must begin by using either Premise or Assumption. If you are trying to prove the validity of an argument, you must start the problem by writing down a vertical line with a horizontal stroke on it. You then list the premises to the immediate right of this vertical line and above the horizontal stroke. The goal is to get the conclusion of the argument listed to the direct right of this vertical line.

ASSUMPTION

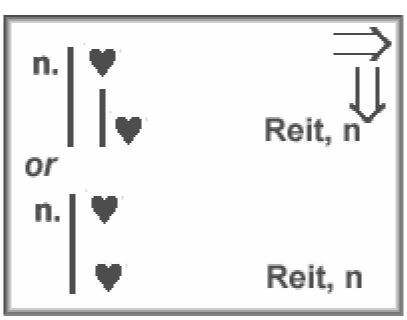
The rule Assumption is both the easiest and hardest rule in the system. Using it is easy. Begin a new vertical line with a horizontal stroke. Place the formula you want to assume to the immediate right of this vertical line, and above the horizontal stroke. This rule is easy because you can assume any formula any time you want. It is hard because you will not have solved the problem until you have the goal or answer listed to the left of all the assumptions you have made. However, only a few of the rules allow you to move to the left of an assumption, thus discharging it. In other words any time you make an assumption you must ultimately discharge it by using a left-moving rule. As a result it is essential to be extremely careful and selective when using the rule Assumption. You should never use Assumption unless you know which left-moving rule you will later be using to discharge that assumption.

ASSUMPTION #2

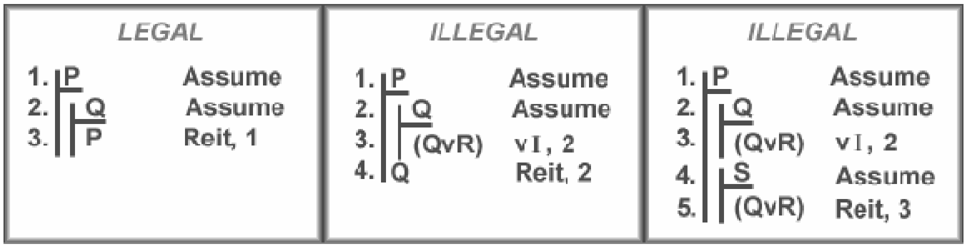


This is another type of assumption. It differs from the rule we call Assumption in one way, however. To use it, stop the last vertical line you drew and start a new vertical line with a horizontal stroke directly under it. Place the formula you want to assume to the right of this vertical line. You will only use the rule Ass#2 when you are constructing a problem that will eventually use the left-moving rule triple-bar introduction or, in some versions of this system, wedge elimination. Normally, this rule is viewed as a variant of Assumption, and so, is justified Assume. We will sometimes follow this practice.

REITERATION

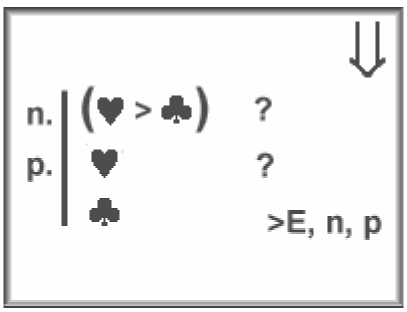


Reiteration is the last of the special rules we will be using. This rule permits us to repeat a formula we have obtained earlier. The rule tells us we can do this if we have not discharged any assumptions we were working under when we first obtained that formula. Compare the examples below to see how we can and cannot use this rule.



THE INTRODUCTION AND ELIMINATION RULES

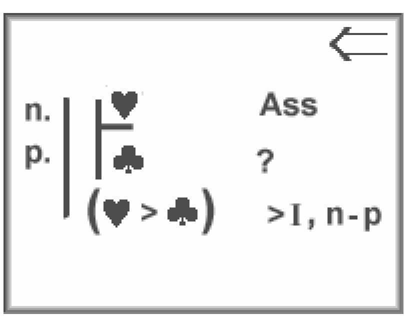
HORSESHOE ELIMINATION



If a derivation contains a formula with a horseshoe as its main connective, and it also contains the left side of that horseshoe claim, you are in luck. The rule, Horseshoe Elimination, permits you to write down the right side of the horseshoe claim. The example below provides illustrations of how this rule works.

1	(P>(Q>R))		Premise
2	(P>Q)		Premise
3	P	/ R	Premise
4	Q		>E, 2, 3
5	(Q>R)		>E, 1, 3
6	R		>E, 4, 5

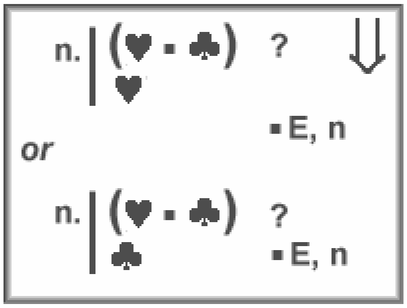
HORSESHOE INTRODUCTION



Horseshoe Introduction is a left-moving rule. It tells us that if we want to create a formula whose main connective is a horseshoe, we should assume its left side. Under this assumption we then need to get the right side of the horseshoe claim we want. Once we have done this we can discharge the assumption we made and write down the horseshoe claim. Study the example below carefully.

1	(P>(Q>R))		Premise
2	(P>Q)	/ (P>R)	Premise
3	P		Assume
4	(Q>R)		>E, 1, 3
5	Q		>E, 2, 3
6	R		>E, 4, 5
7	(P>R)		>I, 3-6

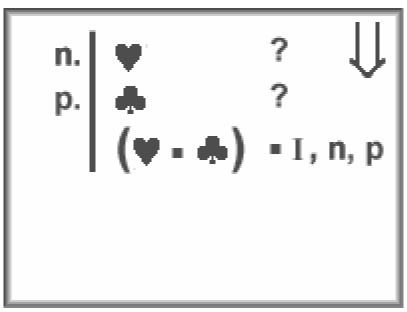
DOT ELIMINATION



Both the introduction and elimination rules for the dot are very easy. The rule Dot Elimination says that if you have a claim whose main connective is a dot, you may write down either side of that claim. Unlike Horseshoe and Triple-Bar Elimination, you don't need to have the other side of the dot claim to do this. A quick example should suffice.

1	(P>(Q.R))		Premise
2	(P. S)	/ R	Premise
3	P		.E, 2
4	(Q.R)		>E, 1, 3
5	R		.E, 4

DOT INTRODUCTION



Dot Introduction is only slightly more complex than Dot Elimination. The rule says you can create a dot claim if you have both of its sides already listed.

1	((P.Q)>(R.S))		Premise
2	(P.T)	/ (Q>(S.T))	Premise
3	Q		Assume
4	P		.E, 2
5	(P.Q)		.I, 3, 4
6	(R.S)		>E, 1, 5
7	S		.E, 6
8	T		.E, 2
9	(S.T)		.I, 7, 8
10	(Q>(S.T))		>I, 3-9

WEDGE INTRODUCTION

n.	♥	?
	(♥v♣)	vI, n
or		
n.	♥	?
	(♣v♥)	vI, n

Wedge Introduction is the easiest rule in the system. To use it, a formula is all you need to have. The rule permits you to create that formula wedge any formula or, any formula wedge that formula. The example below should illustrate this.

1	P		Premise
2	((Rv(PvQ))>S)	/ (M>S)	Premise
3	M		Assume
4	(PvQ)		vI, 1
5	(Rv(PvQ))		vI, 4
6	S		>E, 2, 5
7	(M>S)		>I, 3-6

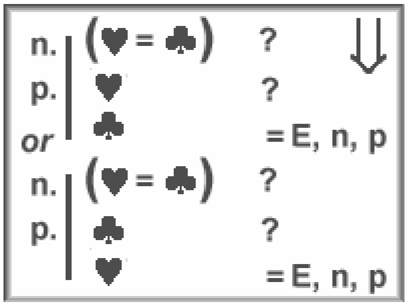
WEDGE ELIMINATION

n.	(♥v♣)	?
p.	♥	Ass
q.	♦	?
r.	♣	Ass #2
s.	♦	?
	♦	vE, n, p-s

Wedge Elimination is quite complex. To use this rule you must have a claim whose main connective is a wedge already listed. Under this formula you need to assume the left side of the wedge, and under this assumption, you need to derive the formula you want to obtain. You then stop the assumption you have been working under and directly under it you begin a second assumption, an assumption that consists in the right side of the wedge claim. You then need to obtain the formula you want to get a second time, but under the second assumption this time. The rule Wedge Elimination then says you can write down the formula you have obtained to the left of this assumption. Study the rule above and the example below carefully before continuing.

1	(P>Q)		Premise
2	(PvR)		Premise
3	((RvS)>T)	/ (QvT)	Premise
4	P		Assume
5	Q		>E, 1, 4
6	(QvT)		vI, 5
7	R		Assume
8	(RvS)		vI, 7
9	T		>E, 3, 8
10	(QvT)		vI, 9
11	(QvT)		vE, 2, 4-10

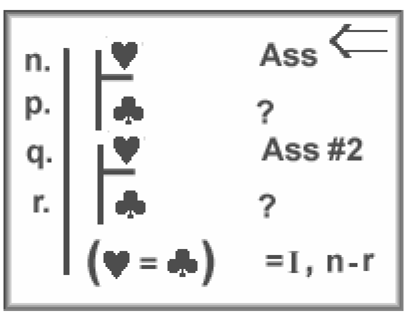
TRIPLE-BAR ELIMINATION



Triple-bar Elimination permits you to write down either side of a triple-bar claim. You can do this if you already have listed both the triple-bar claim and its other side. This rule obviously resembles the rule Horseshoe Elimination. The brief example below should suffice.

1	(P=(Q=(RvS)))		Premise
2	(P.S)	/ Q	Premise
3	P		.E, 2
4	(Q=(RvS))		=E, 1, 3
5	S		.E, 2
6	(RvS)		vI, 5
7	Q		=E, 4, 6

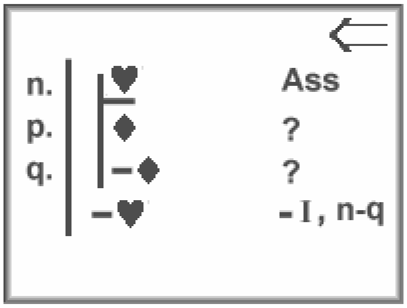
TRIPLE-BAR INTRODUCTION



A more complex rule than Triple-bar Elimination, and another of our left-moving rules, is Triple-bar Introduction. To use this rule, we need to begin by assuming the left side of the triple-bar claim we ultimately want to obtain. Under this assumption we must derive the right side of the triple-bar claim. After we have done this we then stop the assumption we have made and start a second assumption directly under it. This time we assume the right side of the triple-bar claim, and under this assumption we derive the left side of the triple-bar claim. The rule then allows us to move to the left of our assumption and write down the triple-bar claim we have been looking for. Study the rule and example carefully.

1	(P>Q)		Premise
2	(Q=S)		Premise
3	((QvR)>(T.P))	/ (P=S)	Premise
4	P		Assume
5	Q		>E, 1, 4
6	S		=E, 2, 5
7	S		Assume
8	Q		=E, 2, 7
9	(QvR)		vI, 8
10	(T.P)		>E, 3, 9
11	P		.E, 10
12	(P=S)		=I, 4-11

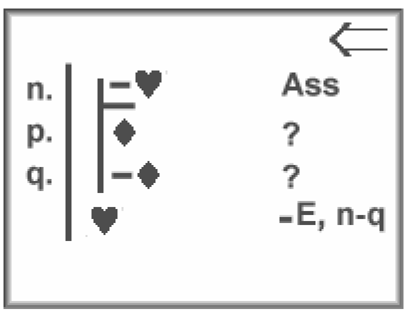
TILDE INTRODUCTION



The rule Tilde Introduction is another left-moving rule. It tells us that if we have made an assumption, and under this assumption we obtained any formula and its negation, we can stop the assumption, move left, and write down a tilde that assumption. This type of rule is sometimes called Reductio ad Absurdum.

1	(P>Q)		Premise
2	((QvR)>-Q)		Premise
3	P	/ -P	Assume
4	Q		>E, 1, 3
5	(QvR)		vI, 4
6	-Q		>E, 2, 5
7	-P		-I, 3-6

TILDE ELIMINATION



Tilde Elimination works just like Tilde Introduction. The only difference is that the assumed formula must begin with a tilde, and the formula we move to the left deletes this tilde. This is another version of Reductio ad Absurdum. For our purposes, it is also important to note that it is a left-moving rule.

1	Q		Premise
2	(-P>-Q)	/ P	Premise
3	-P		Assume
4	Q		Reit, 1
5	-Q		>E, 2, 3
6	P		-E, 3-5

PROOFS FOR BREAKFAST

OVER EASY

((P > - Q) > (R > Q))
- Q

- R

POACHED

(P . Q)
(((R > P) . (P = Q)) > S)

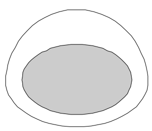
((S v T) . (- B v S))

FRIED

((P = - Q) . Q)
(P v R)

(S > R)

HARD BOILED



(P = Q)
((- P = - Q) > (- P v - Q))

(- P . - Q)

THE OMELET FROM HELL

(((P = Q) > (- P v Q)) > (R > S))

(- R v S)