$$\text{(CONSEQUENCE)}\quad \frac{\begin{array}{ll} c:(P',R',G',Q') & \\ R \subseteq R' & G' \subseteq G \\ P \Rightarrow P' & Q' \Rightarrow Q \end{array}}{c:(P,R,G,Q)}$$

Außerdem gibt es Regeln für Disjunktion und Existenzquantoren:

$$\text{(DISJ)}\quad \frac{c:(P_1,R,G,Q) \quad c:(P_2,R,G,Q)}{c:(P_1 \vee P_2,R,G,Q)} \qquad \text{(EX)}\quad \frac{x \notin fv(c,R,G,Q) \quad c:(P,R,G,Q)}{c:(\exists x.\, P,R,G,Q).}$$

Wir haben auch die Frame-Rule.
Allerdings könnte die Frame den shared State nicht beschreiben.
Diese Beschreibung muss interferenzfrei sein,
sowohl von Seiten der Umgebung als auch von Seite des Programms.

$$\text{(FRAME)}\quad \frac{\begin{array}{l} c:(P,R,G,Q) \\ stable(P',R \cup G) \\ modifies(c) \cap fv(P') = \emptyset \end{array}}{c:(P * P', R, G, Q * P').}$$

Die Beweisregeln für Konstrukte der Programmiersprache
verhalten sich, wie erwartet.
Wir haben keine Regel für while
sondern modellieren Schleifen als:

$$\underline{\text{while}}\ b\ \underline{\text{do}}\ c\ \underline{\text{od}} := (assume(b); c)^*; assume(\neg b).$$

Analog ist

if b then c_1 else c_2 fi $:=$ (assume (b); c_1) + (assume $(\neg b)$; c_2).

$$(\text{Ship})\ \frac{\text{stable}(P,R)}{\text{ship}:(P,R,G,P)} \qquad \frac{\text{stable}(P,R) \quad c:(P,R,G,P)}{c^*:(P,R,G,P)}\ (\text{LOOP})$$

$$(\text{SEQ})\ \frac{c_1:(P,R,G,P') \quad c_2:(P',R,G,Q)}{c_1;c_2:(P,R,G,Q)} \qquad \frac{c_1:(P,R,G,Q) \quad c_2:(P,R,G,Q)}{c_1+c_2:(P,R,G,Q)}\ (\text{CHOICE})$$

Für primitive Befehle greifen wir zurück auf die unterliegende Separation-Logik:

$$(\text{COM})\ \frac{\vdash_{SL} \{P\}\ com\ \{Q\} \qquad \text{modifies}(com) \cap \text{free}(R,G) = \emptyset}{com:(P,R,G,Q).}$$

Die Regel für die Parallelkomposition ist wie bei Rely-Guarantee, nur dass wir jetzt $*$ nehmen:

$$(\text{PAR})\ \frac{\begin{array}{ll} c_1:(P_1, R \cup G_2, G_1, Q_1) & \text{modifies}(c_1) \cap \text{free}(c_2,P_2,Q_2)=\emptyset \\ c_2:(P_2, R \cup G_1, G_2, Q_2) & \text{modifies}(c_2) \cap \text{free}(c_1,P_1,Q_1)=\emptyset \end{array}}{c_1 \| c_2 : (P_1 * P_2,\ R,\ G_1 \cup G_2,\ Q_1 * Q_2).}$$

Die schwierigste Regel ist jetzt atomic c.

Zwei Schritte.

Prüfe zunächst, ob der atomare Block seine Spezifikation in leerer Umgebung erfüllt. (nicht einfach)

Prüfe dann, dass Pre- und Postcondition stabil gegenüber Rely.

Einfacher Teil:

$$\text{(ATOMN)}\ \frac{\underline{\text{atomic}}\ c : (P, \varnothing, G, Q) \qquad \text{stable}(P, R) \qquad \text{modifies}(c) \cap \text{free}(R) = \varnothing \qquad \text{stable}(Q, R)}{\underline{\text{atomic}}\ c : (P, R, G, Q)}$$

Schwierigere Teil:

1. Versuch:

$$\frac{c : (A * C, \varnothing, \varnothing, B * D) \qquad (A \rightsquigarrow B) \subseteq G \qquad \text{modifies}(c) \cap \text{free}(G) = \varnothing}{\underline{\text{atomic}}\ c : (\boxed{A} * C, \varnothing, G, \boxed{B} * D)}$$

Im atomaren Block dürfen wir den shared state $\boxed{A}$ zugreifen, müssen aber sicherstellen, dass die Änderung von A zu B von G gefasst ist. In der Praxis zu stark, man ändert den gesamten shared state.

2. Versuch:

$$\text{(FATOM)}\ \frac{c : (A * C, \varnothing, \varnothing, B * D) \qquad A \rightsquigarrow B \subseteq G \qquad \underline{\text{modifies}(c)} \cap \underline{\text{free}(G)} = \varnothing}{\underline{\text{atomic}}\ c : (\underline{\boxed{A * F}} * C, \varnothing, G, \underline{\boxed{B * F}} * D).}$$

Zusammen:

$$\text{(ATOMIC)}\ \frac{\begin{array}{l} A' \rightsquigarrow B' \subseteq G \qquad c : (A' * C, \varnothing, \varnothing, B' * D) \\ \bar{y} \cap \text{free}(c, C, G, B, D) \qquad A \Rightarrow A' * F \qquad B' * F \Rightarrow B \\ \text{modifies}(c) \cap \text{free}(R, G) = \varnothing \qquad \text{sem\_stable}(\exists \bar{y}.\, A, R) \qquad \text{sem\_stable}(B, R) \end{array}}{\underline{\text{atomic}}\ c : (\underline{\boxed{\exists \bar{y}.\, A}} * C, R, G, \exists \bar{y}.\, \boxed{B} * D)}$$

$$\text{(ATOM)}\ \frac{}{c : (A' * C,\ \emptyset,\ \emptyset,\ B' * D)}\ (1)$$

$$\text{(CONSEQUENCE)}\ \frac{\text{atomic } c : (\boxed{A' * F} * C,\ \emptyset,\ G,\ \boxed{B' * F} * D)}{\text{atomic } c : (\boxed{A} * C,\ \emptyset,\ G,\ \boxed{B} * D)}\ (2)$$

$$\text{(EX)}\ \frac{\text{atomic } c : (\boxed{A} * C,\ \emptyset,\ G,\ \boxed{B} * D)}{\text{atomic } c : (\exists \bar{y}.\ \boxed{A} * C,\ \emptyset,\ G,\ \boxed{B} * D)}\ (3)$$

$$\text{(CONSEQ)}\ \frac{\text{atomic } c : (\exists \bar{y}.\ \boxed{A} * C,\ \emptyset,\ G,\ \boxed{B} * D)}{\text{atomic } c : (\boxed{\exists \bar{y}.\ A} * C,\ \emptyset,\ G,\ \exists \bar{y}.\ \boxed{B} * D)}\ (4)$$

$$\text{(ATOM2)}\ \frac{\text{atomic } c : (\boxed{\exists \bar{y}.\ A} * C,\ \emptyset,\ G,\ \exists \bar{y}.\ \boxed{B} * D)}{\text{atomic } c : (\boxed{\exists \bar{y}.\ A} * C,\ R,\ G,\ \exists \bar{y}.\ \boxed{B} * D)}\ (5)$$

(1) $\text{modifies}(c) \cap \text{fv}(G) = \emptyset$

$(A' \rightsquigarrow B') \in G$

(2) $A \Rightarrow A' * F$, $B' * F \Rightarrow B$

(3) $\text{fv}(c, G, B, D) \cap \bar{y} = \emptyset$

(4) $\text{free}(C) \cap \bar{y} = \emptyset$

(5) sem slash $\exists x$, $\text{modifies}(c) \cap \text{fv}(R) = \emptyset$.