09-0: **DFAs & regular expressions**

- Regular expressions are *string generators* – they tell us how to generate all strings in a language L
- Finite Automata (DFA, NFA) are *string acceptors* – they tell us if a specific string w is in L
- CFGs are string generators
- Are there string acceptors for Context-Free languages?

09-1: **DFAs & regular expressions**

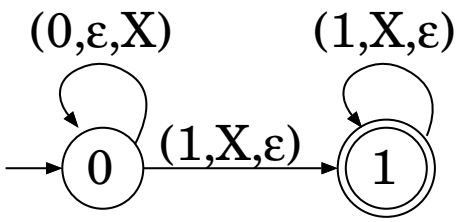
- Regular expressions are *string generators* – they tell us how to generate all strings in a language L
- Finite Automata (DFA, NFA) are *string acceptors* – they tell us if a specific string w is in L
- CFGs are string generators
- Are there string acceptors for Context-Free languages?
- YES! Push-down automata

09-2: **Push-Down Automata**

- DFA could not accept languages such as 0^n1^n because they have no memory
- We can give an NFA memory – stack
  - Examine the next symbol in the input, and pop off the top symbol(s) on the stack
  - Transition to the next state, depending upon what the next symbol in the input is, and what the top of the stack is, and (potentially) push more symbols on the top of the stack
  - Accept if you end up in an accept state *with an empty stack*

09-3: **Push-Down Automata**

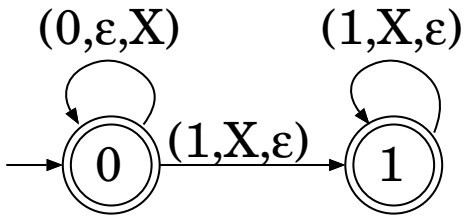
- PDA for $L = \{0^n1^n | n > 0\}$



What if we wanted $n \geq 0$?

09-4: **Push-Down Automata**

- PDA for $L = \{0^n1^n | n \geq 0\}$



09-5: **Push-Down Automata**

$M = (K, \Sigma, \Lambda, \Delta, s, F)$

- K set of states (just like DFA)
- Σ alphabet (just like DFA)
- Λ stack alphabet (symbols that can be pushed on the stack)
- $\Delta \subseteq (K \times (\Sigma \cup \{\epsilon\}) \times \Lambda^*) \times (K \times \Lambda^*)$ Transition relation
- $s \in K$ Initial state
- $F \subseteq K$ Final state(s)

PDA accepts w if we end up in a final state with an empty stack

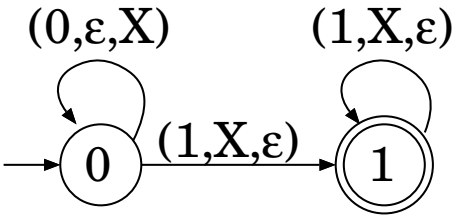
09-6: **Push-Down Automata**

- Transition Relation
  - All PDAs are non-deterministic (for now!)
  - If $((q_1, a, \lambda_1), (q_2, \lambda_2))$ is in Δ
    - In state q_1
    - Next symbol in the input is a
    - Top of the stack is λ_1
    - We can pop λ_1 off the top of the stack, consume a, transition to q_2, and push λ_2 on the top of the stack

09-7: **Push-Down Automata**

- PDA Configuration
  - (<state>, <remaining string>, <stack>)
    - Top of the stack on the left

09-8: **Push-Down Automata**



$$
\begin{aligned}
(q_0, 000111, \epsilon) &\mapsto_M (q_0, 00111, X) \\
&\mapsto_M (q_0, 0111, XX) \\
&\mapsto_M (q_0, 111, XXX) \\
&\mapsto_M (q_1, 11, XX) \\
&\mapsto_M (q_1, 1, X) \\
&\mapsto_M (q_1, \epsilon, \epsilon)
\end{aligned}
$$

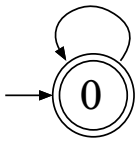
09-9: **Push-Down Automata**

- Create a PDA for all strings over {a, b} with the same number of a's as b's

09-10: **Push-Down Automata**

- Create a PDA for all strings over {a, b} with the same number of a's as b's

(a,ε,A)
(b,A,ε)
(b,ε,B)
(a,B,ε)



09-11: **Push-Down Automata**

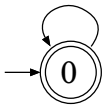
- Create a PDA for all strings over {a, b} with the same number of a's as b's
- We can also describe a PDA by listing Δ, s, and F

(a,ε,A)
(b,A,ε)
(b,ε,B)
(a,B,ε)

09-12: **Push-Down Automata**

$((q_0, a, \epsilon), (q_0, A))$
$((q_0, b, A), (q_0, \epsilon))$
$((q_0, b, \epsilon), (q_0, B))$
$((q_0, a, B), (q_0, \epsilon))$

$S = q_0, F = \{q_0\}$

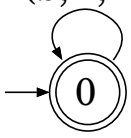


- Create a PDA for all strings over {a, b} with twice as many a's as b's

09-13: **Push-Down Automata**

- Create a PDA for all strings over {a, b} with twice as many a's as b's

(a,ε,A)
(a,B,ε)
(b,AA,ε)
(b,ε,BB)
(b,A,B)

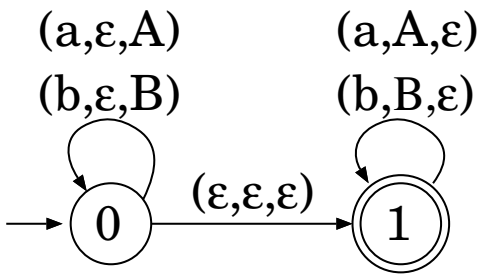


09-14: **Push-Down Automata**

- Create a PDA for $L = \{ww^R : w \in \{a, b\}*\}$

09-15: **Push-Down Automata**

- Create a PDA for $L = \{ww^R : w \in \{a, b\}*\}$

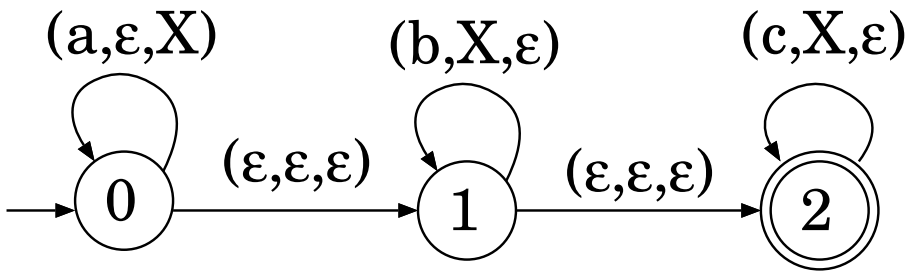


09-16: **Push-Down Automata**

- Create a PDA for $L = \{a^i b^j c^k : i = j + k\}$

09-17: **Push-Down Automata**

- Create a PDA for $L = \{a^i b^j c^k : i = j + k\}$

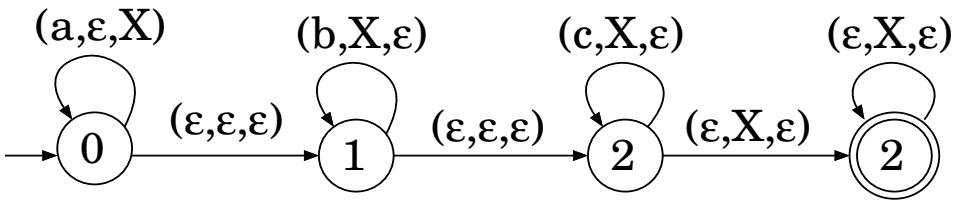


09-18: **Push-Down Automata**

- Create a PDA for $L = \{a^i b^j c^k : i > j + k\}$

09-19: **Push-Down Automata**

- Create a PDA for $L = \{a^i b^j c^k : i > j + k\}$

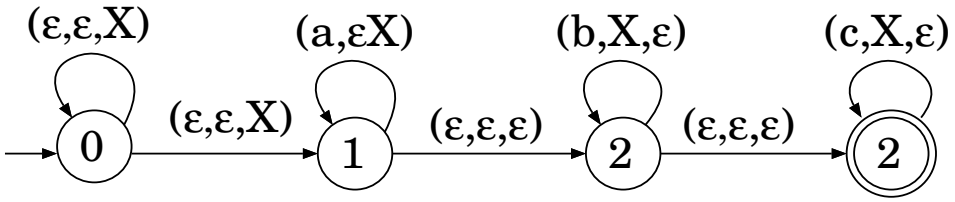


09-20: **Push-Down Automata**

- Create a PDA for $L = \{a^i b^j c^k : i < j + k\}$

09-21: **Push-Down Automata**

- Create a PDA for $L = \{a^i b^j c^k : i < j + k\}$



09-22: **PDA Languages**

The Push-Down Automata Languages, L_{PDA}, is the set of all languages that can be described by some PDA:

- $L_{PDA} = \{L : \exists \text{ PDA } M \wedge L[M] = L\}$

We already know $L_{PDA} \supset L_{DFA}$ (why)?

09-23: **PDA Languages**

The Push-Down Automata Languages, L_{PDA}, is the set of all languages that can be described by some PDA:

- $L_{PDA} = \{L : \exists \text{ PDA } M \wedge L[M] = L\}$

We already know $L_{PDA} \supset L_{DFA}$ – every DFA is just a PDA that ignores the stack.

- $L_{CFG} \subseteq L_{PDA}$?
- $L_{PDA} \subseteq L_{CFG}$?

09-24: $L_{CFG} \subseteq L_{PDA}$

- Given any CFG G, can we create a PDA M such that $L[M] = L[G]$?
  - Idea: We will first use the stack to derive the input string, and then check to make sure our derivation matches the input

09-25: $L_{CFG} \subseteq L_{PDA}$

Given any CFG $G = (V, \Sigma, R, S)$, can we create a PDA $M = (K, \Sigma', \Gamma, \Delta, s, F)$ as follows:

- $K = \{q_1, q_2\}$
- $\Sigma' = \Sigma$
- $\Gamma = V$
- $\Delta =$
  $((q_1, \epsilon, \epsilon), (q_2, S))$
  $((q_2, \epsilon, N), (q_2, X))$ For each rule $(N, X) \in R$
  $((q_2, a, a), (q_2, \epsilon))$ For each $a \in \Sigma$
- $s = q_1$
- $F = \{q_2\}$

09-26: $L_{CFG} \subseteq L_{PDA}$

Example:
$S \to aSa$
$S \to bSb$
$S \to c$

(what language does this CFG generate?)

09-27: $L_{CFG} \subseteq L_{PDA}$

Example:

$S \to aSa$ $\quad L = \{wcw^R : w \in \{a,b\}^*\}$

$S \to bSb$

$S \to c$

$((q_0, \epsilon, \epsilon), (q_1, S))$

$((q_1, \epsilon, S), (q_1, aSa))$

$((q_1, \epsilon, S), (q_1, bSb))$

$((q_1, \epsilon, S), (q_1, c))$

$((q_1, a, a), (q_1, \epsilon))$

$((q_1, b, b), (q_1, \epsilon))$

$((q_1, c, c), (q_1, \epsilon))$

09-28: $L_{CFG} \subseteq L_{PDA}$

- First, we will show that any PDA can be converted to an equivalent machine with the following restriction:
- All transitions are of the form $(q_1, A, B) \to (q_2, C)$, where:
  - $A \in \Sigma \cup \{\epsilon\}$
  - $B \in \Gamma$
  - $C \in \Gamma^*$, with $|C| \leq 2$

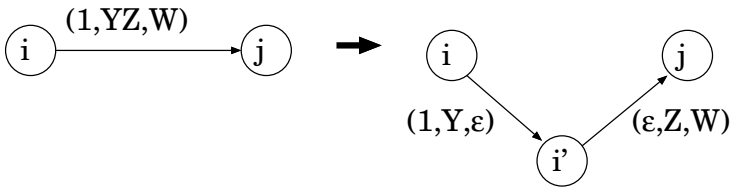
09-29: $L_{CFG} \subseteq L_{PDA}$

- First, we will show that any PDA can be converted to an equivalent machine with the following restriction:
- All transitions are of the form $(q_1, A, B) \to (q_2, C)$, where:
  - $A \in \Sigma \cup \{\epsilon\}$
  - $B \in \Gamma$
  - $C \in \Gamma^*$, with $|C| \leq 2$
- How can we remove all transitions where $B \geq 2$?

09-30: $L_{CFG} \subseteq L_{PDA}$

- All transitions are of the form $(q_1, A, B) \to (q_2, C)$
- How can we remove all transitions where $B \geq 2$?

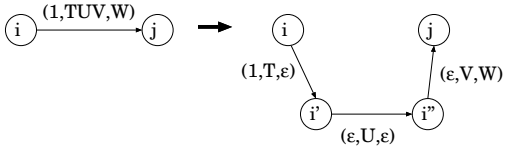
$$((q_i, 1, YZ), (q_j, W)) \to \begin{array}{l} ((q_i, 1, X), (q_i', \epsilon)) \\ ((q_i', \epsilon, Z), (q_j, W) \end{array}$$



09-31: $L_{CFG} \subseteq L_{PDA}$

- All transitions are of the form $(q_1, A, B) \rightarrow (q_2, C)$
- How can we remove all transitions where $B \geq 2$?

$$\begin{aligned}((q_i, 1, TUV), (q_j, W)) \rightarrow \quad & ((q_i, 1, T), (q_i', \epsilon)) \\ & ((q_i', \epsilon, U), (q_i'', \epsilon)) \\ & ((q_i'', \epsilon, V), (q_j, W))\end{aligned}$$



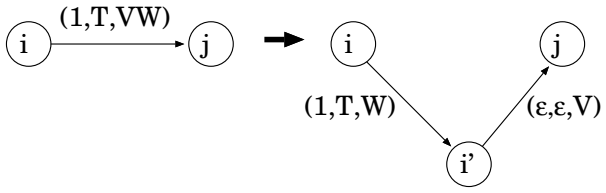
09-32: $L_{CFG} \subseteq L_{PDA}$

- First, we will show that any PDA can be converted to an equivalent machine with the following restriction:
- All transitions are of the form $(q_1, A, B) \rightarrow (q_2, C)$, where:
  - $A \in \Sigma \cup \{\epsilon\}$
  - $B \in \Gamma$
  - $C \in \Gamma^*$, with $|C| \leq 2$
- How can we remove all transitions where $C \geq 2$? (stronger than we need – see why in a moment)

09-33: $L_{CFG} \subseteq L_{PDA}$

- How can we remove all transitions where $C \geq 2$?

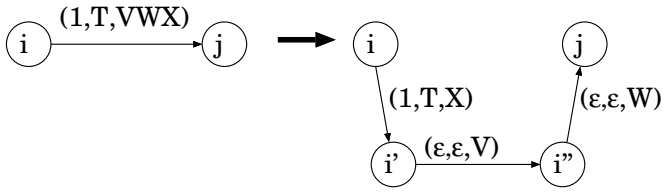
$$\begin{aligned}((q_i, 1, T), (q_j, VW)) \Rightarrow \quad & ((q_i, 1, T), (q_i', W)), \\ & ((q_i', \epsilon, \epsilon), (q_j, V))\end{aligned}$$



09-34: $L_{CFG} \subseteq L_{PDA}$

- How can we remove all transitions where $C \geq 2$?

$$\begin{aligned}((q_i, 1, T), (q_j, VWX)) \Rightarrow \quad & ((q_i, 1, T), (q_i', X)), \\ & ((q_i', \epsilon, \epsilon), (q_i'', W)) \\ & ((q_i'', \epsilon, \epsilon), (q_j, V))\end{aligned}$$



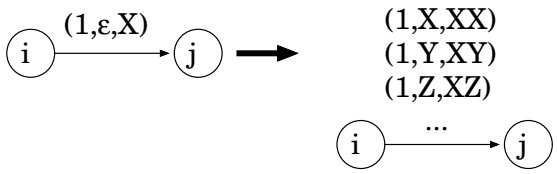
09-35: $L_{CFG} \subseteq L_{PDA}$

- First, we will show that any PDA can be converted to an equivalent machine with the following restriction:
- All transitions are of the form $(q_1, A, B) \rightarrow (q_2, C)$, where:
  - $A \in \Sigma \cup \{\epsilon\}$
  - $B \in \Gamma$
  - $C \in \Gamma^*$, with $|C| \leq 2$
- How can we remove all transitions where $B = \epsilon$?
  - Exceptions?

09-36: $L_{CFG} \subseteq L_{PDA}$

- How can we remove all transitions where $B = \epsilon$?

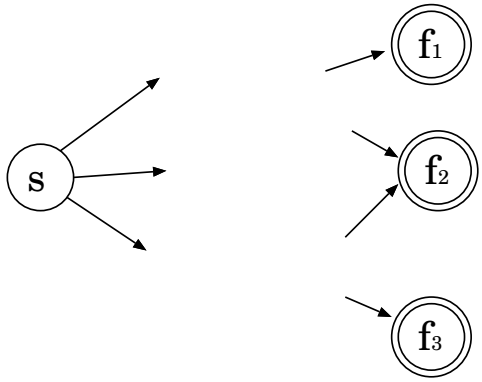
$$\begin{aligned}((q_i, 1, \epsilon), (q_j, X)) \Rightarrow \quad & ((q_i, 1, X), (q_j, XX)), \\ & ((q_i, 1, Y), (q_j, XY)), \\ & ((q_i, 1, Z), (q_j, XZ)), \\ & \ldots\end{aligned}$$



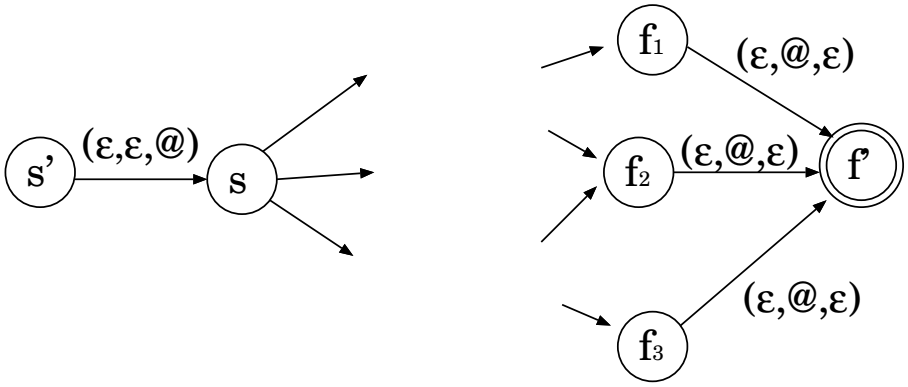
09-37: $L_{CFG} \subseteq L_{PDA}$

- What about when the stack is empty?
  - First, push a special "end of stack symbol" on top of stack
  - Pop this symbol off at the end

09-38: $L_{CFG} \subseteq L_{PDA}$



09-39: $L_{CFG} \subseteq L_{PDA}$



09-40: $L_{CFG} \subseteq L_{PDA}$

- All transitions in the PDA are now of the form $(q_1, A, B) \rightarrow (q_2, C)$, where:
  - $A \in \Sigma \cup \{\epsilon\}$
  - $B \in \Gamma$
  - $C \in \Gamma^*$, with $|C| \leq 2$
- We can now create a CFG equivalent to this PDA

09-41: $L_{CFG} \subseteq L_{PDA}$

- All non-terminals will be of the form:
- $< q_i, A, q_j >$

  This non-terminal will generate all strings that will drive the PDA from state q_i to state q_j, while popping A off the top of the stack
- $< q_i, \epsilon, q_j >$

  This non-terminal will generate all strings that will drive the PDA from state q_i to state q_j, leaving the stack unchanged.

09-42: $L_{CFG} \subseteq L_{PDA}$

- For each transion of the form:

$$((q, a, B), (r, C))$$

add the rule:

$$< q, B, p > \rightarrow a < r, C, p >$$

for all states p

- $B \in \Gamma, C \in \Gamma \cup \{\epsilon\}$

09-43: $L_{CFG} \subseteq L_{PDA}$

- For each transion of the form:

$$((q, a, B), (r, C_1 C_2))$$

add the rule:

$$< q, B, p > \rightarrow a < r, C_1, p' >< p', C_2, p >$$

for all pairs of states p, p'

- $C_1, C_2, B \in \Gamma$, $a \in \Sigma \cup \epsilon$

09-44: $L_{CFG} \subseteq L_{PDA}$

- For each state q, add the rule:

$$< q, \epsilon, q > \rightarrow \epsilon$$

- Start state

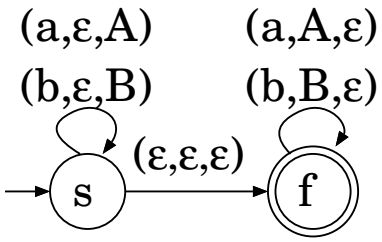
$$< s, @, f' >$$

09-45: $L_{CFG} \subseteq L_{PDA}$

- Example: Create PDA for $L = \{ww^R : w \in \{a, b\}^*\}$

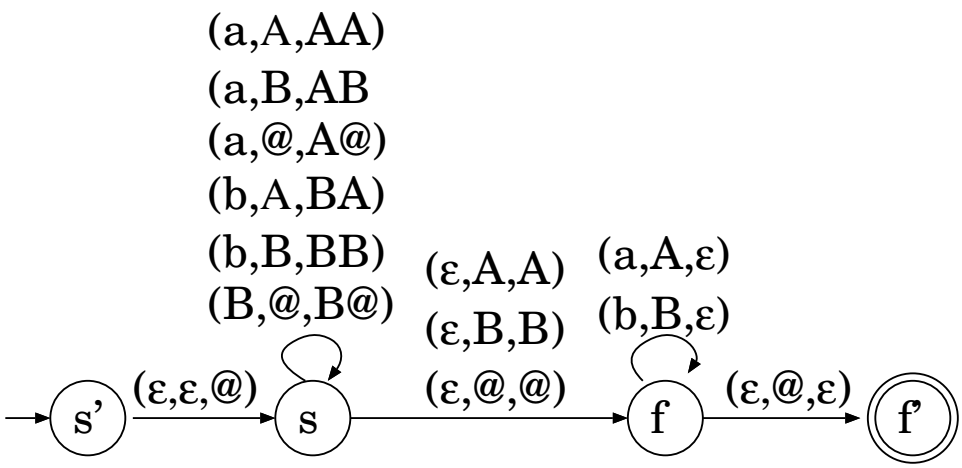
09-46: $L_{CFG} \subseteq L_{PDA}$

- Example: Create PDA for $L = \{ww^R : w \in \{a, b\}^*\}$



09-47: $L_{CFG} \subseteq L_{PDA}$

- Example: Create PDA for $L = \{ww^R : w \in \{a, b\}^*\}$



09-48: $L_{CFG} \subseteq L_{PDA}$

$((s,a,A),(s,AA))$	$((s,a,B),(s,AB))$	$((s,a,@),(s,A@))$
$((s,b,A),(s,BA))$	$((s,b,B),(s,BB))$	$((s,b,@),(s,B@))$
$((s,\epsilon,A),(f,A))$	$((s,\epsilon,B),(f,B))$	$((s,\epsilon,@),(f,@))$
$((f,a,a),(f,\epsilon))$	$((f,b,b),(f,\epsilon))$	$((f,\epsilon,@),(f',\epsilon))$

09-49: $L_{CFG} \subseteq L_{PDA}$

$$((f,a,A),(f,\epsilon))$$

$< f,A,s > \rightarrow a < f,\epsilon,s >$
$< f,A,f > \rightarrow a < f,\epsilon,f >$
$< f,A,f' > \rightarrow a < f,\epsilon,f' >$
$< f,A,s' > \rightarrow a < f,\epsilon,s' >$

09-50: $L_{CFG} \subseteq L_{PDA}$

$$((f,\epsilon,@),(f',\epsilon))$$

$< f,@,s > \rightarrow < f',\epsilon,s >$
$< f,@,f > \rightarrow < f',\epsilon,f >$
$< f,@,f' > \rightarrow < f',\epsilon,f' >$
$< f,@,s' > \rightarrow < f',\epsilon,s' >$

09-51: $L_{CFG} \subseteq L_{PDA}$

$$((s,a,B),(s,AB))$$

$< s,B,s > \rightarrow a < s,A,s >< s,B,s >$	$< s,B,s > \rightarrow a < s,A,s' >< s',B,s >$
$< s,B,s > \rightarrow a < s,A,f >< f,B,s >$	$< s,B,s > \rightarrow a < s,A,f' >< f',B,s >$
$< s,B,f > \rightarrow a < s,A,s >< s,B,f >$	$< s,B,f > \rightarrow a < s,A,s' >< s',B,f >$
$< s,B,f > \rightarrow a < s,A,f >< f,B,f >$	$< s,B,f > \rightarrow a < s,A,f' >< f',B,f >$
$< s,B,s' > \rightarrow a < s,A,s >< s,B,s' >$	$< s,B,s' > \rightarrow a < s,A,s' >< s',B,s' >$
$< s,B,s' > \rightarrow a < s,A,f >< f,B,s' >$	$< s,B,s' > \rightarrow a < s,A,f' >< f',B,s' >$
$< s,B,f' > \rightarrow a < s,A,s >< s,B,f' >$	$< s,B,f' > \rightarrow a < s,A,s' >< s',B,f' >$
$< s,B,f' > \rightarrow a < s,A,f >< f,B,f' >$	$< s,B,f' > \rightarrow a < s,A,f' >< f',B,f' >$