COURSE: THEORY OF AUTOMATA COMPUTATION

TOPICS TO BE COVERED

- Pushdown Automata and Context-Free Languages

NPDAS

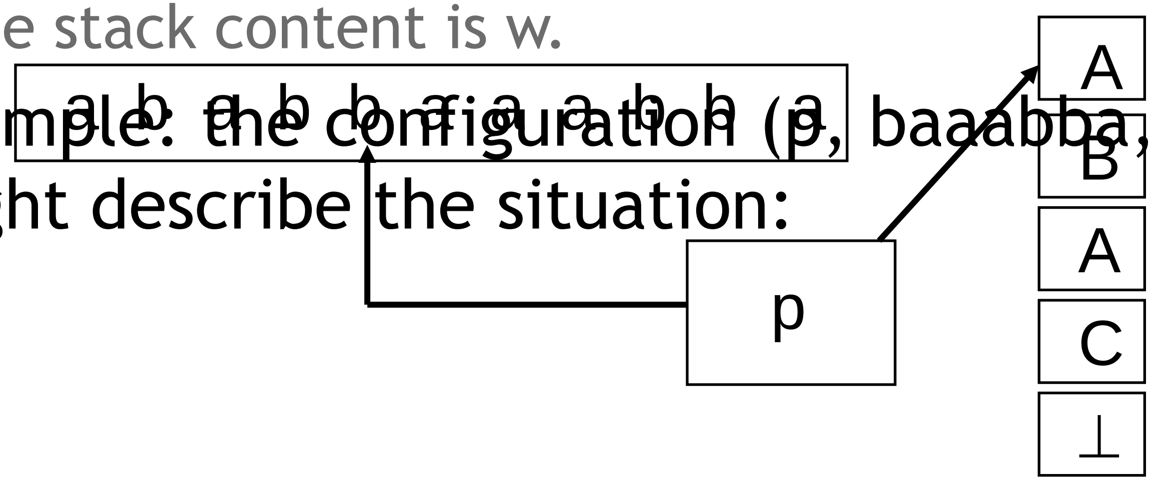
- A NPDA (Nondeterministic PushDown Automata) is a 7-tuple $M = (Q,\Sigma,\Gamma, \delta ,s, \perp , F)$ where
  - Q is a finite set (the states)
  - Σ is a finite set (the input alphabet)
  - Γ is a finite set (the stack alphabet)
  - $\delta \subseteq (Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma) \times (Q \times \Gamma^*)$ is the transition relation
  - $s \in Q$ is the start state
  - $\perp \in \Gamma$ is the initial stack symbol
  - $F \subseteq Q$ is the final or accept states
- $((p,a,A),(q,B_1B_2\ldots B_k)) \in \delta$ means that

whenever the machine is in state p reading input symbol a on the input tape and A on the top of the stack, it pops A off the stack, push $B_1B_2\ldots B_k$ onto the stack (B_k first and B_1 last), move its read head right one cell past the one storing a and enter state q.

$((p,\varepsilon,A),(q,B_1B_2\ldots B_k)) \in \delta$ means similar to $((p,a,A),(q,B_1B_2\ldots B_k)) \in \delta$ except that it need not scan and consume any input symbol.

CONFIGURATIONS

- Collection of information used to record the snapshot of an executing NPDA
- an element of $Q \times \Sigma^* \times \Gamma^*$.
- Configuration $C = (q, x, w)$ means
  - the machine is at state q,
  - the rest unread input string is x,
  - the stack content is w.
- Example: the configuration $(p, baaabba, ABAC\perp)$ might describe the situation:



START CONFIGURATION AND THE NEXT CONFIGURATION RELATIONS

- Given a NPDA M and an input string x, the configuration (s, x, $\perp$) is called the start configuration of NPDA on x.
- $CF_M =_{def} Q \times \Sigma^* \times \Gamma^*$ is the set of all possible configurations for a NPDA M.
- One-step computation ($-->_M$) of a NPDA:
  - (p, ay, Aβ) $-->_M$ (q, y , $\gamma\beta$) for each ((p,a,A), (q, γ)) $\in \delta$. (1)
  - (p, y, Aβ) $-->_M$ (q, y, $\gamma\beta$) for each ((p,ε,A),(q, γ)) $\in \delta$. (2)
  - Let the next configuration relation $-->_M$ on CF_M^2 be the set of pairs of configurations satisfying (1) and (2).
  - $-->_M$ describes how the machine can move from one configuration to another in one step. (i.e., C $-->_M$ D iff D can be reached from C by executing one instruction)
  - Note: NPDA is nondeterministic in the sense that for each C there may exist multiple D's s.t. C $-->_M$ D.

MULTI-STEP COMPUTATIONS AND ACCEPTANCE

- Given a next configuration relation $-->_M$:
  Define $--->^n_M$ and $--->^*_M$ as usual, i.e.,
  - $C -->^0_M D$ iff $C = D$.
  - $C -->^{n+1}_M$ iff $\exists E\ C-->^n_M E$ and $E-->_M D$.
  - $C -->^*_M D$ iff $\exists n \geq 0\ C -->^n_M D$.
  - i.e., $--->^*_M$ is the ref. and trans. closure of $-->_M$.
- Acceptance: When will we say that an input string x is accepted by an NPDA M?
  - two possible answers:
  - 1. by final states: M accepts x (by final state) iff
  - $(s,x,\perp) -->^*_M (p,\varepsilon,\alpha)$ for some final state $p \in F$.
  - 2. by empty stack: M accepts x by empty stack iff
  - $(s,x,\perp) -->^*_M (p,\varepsilon,\varepsilon)$ for any state p.
  - Remark: both kinds of acceptance have the same expressive power.

LANGUAGE ACCEPTED BY A NPDAS

$M = (Q,\Sigma,\Gamma,\delta,s,,F)$: a NPDA.

The languages accepted by M is defined as follows:

- 1. accepted by final state:
- $L_f(M) = \{x \mid M$ accepts x by final state$\}$
- 2. accepted by empty stack:
- $L_e(M) = \{x \mid M$ accepts x by empty stack$\}$.
- 3. Note: Depending on the context, we may sometimes use L_f and sometimes use L_e as the official definition of the language accepted by a NPDA. I.e., if there is no worry of confusion, we use L(M) instead of $L_e(M)$ or $L_f(M)$ to denote the language accepted by M.
- 4. In general $L_e(M) \neq L_f(M)$.

SOME EXAMPLE NPDAS

Ex 23.1 : M_1: A NPDA accepting the set of balanced strings of parentheses [ ] by empty stack.

- M_1 requires only one state q and behaves as follows:

1. while input is '[' : push '[' onto the stack ;
2. while input is ']' and top is '[' : pop
3. while input is 'ε' and top is $\perp$: pop.

Formal definition: $Q = \{q\}$, $\Sigma = \{[,]\}$, $\Gamma = \{[, \perp\}$,

start state = q, initial stack symbol = $\perp$.

$\delta = \{$ ((q,[, $\perp$), (q, [$\perp$)), ((q,[, [), (q, [[)), // 1

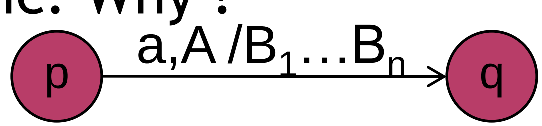
((q,], [), (q, ε)), // 2

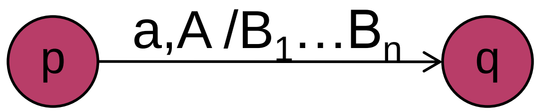
((q,ε, $\perp$), (q, ε)) $\}$ // 3

Transition Diagram representation of the program δ :

$((p,a\ A)\ ,\ (q,B_1 \dots B_n)) \in \delta$ =>

- This machine is not deterministic. Why ?





p
a,A /B1…Bn
q

EXAMPLE : EXECUTION SEQUENCES OF M1

- let input x = [[[ ]] [ ]] [ ]. Then below is a successful computation of M_1 on x:
- (q, [[[ ]] [ ]] [ ], ⊥) : the start configuration
  $-->_M$ (q, [[ ]] [ ]] [ ], [⊥) instruction or transition (i)
  $-->_M$ (q, [ ]] [ ]] [ ], [[⊥) transition (ii)
  $-->_M$ (q,]] [ ]] [ ], [[[⊥) transition (ii)
  $-->_M$ (q,] [ ]] [ ], [[⊥) transition (iii)
  $-->_M$ (q, [ ]] [ ], [⊥) transition (iii)
  $-->_M$ (q,]] [ ], [[⊥) transition (ii)
  $-->_M$ (q,] [ ], [⊥) transition (iii)
  $-->_M$ (q, [ ], ⊥) transition (iii)
  $-->_M$ (q,], [⊥) transition (i)
  $-->_M$ (q, , ⊥) transition (iii)
  $-->_M$ (q, ,) transition (iv)
  accepts by empty stack

FAILURE COMPUTATION OF M1 ON X

- Note besides the above successful computation, there are other computations that fail.

Ex: (q, [[[ ]] [ ]] [ ], $\perp$) : the start configuration

--> $^*_M$ (q, [ ], $\perp$)

--> $_M$ (q, [ ],) transition (iv)

a dead state at which the input is not empty and we cannot move further ==> failure!!

Note: For a NPDA to accept a string x, we need *only one successful computation* (i.e., $\exists$ D = (_, ε, ε) with empty input and stack s.t. (s,x,$\perp$) --> $^*_M$ D.)

- Theorem 1: String x $\in$ {[,]}* is balanced iff it is accepted by M_1 by empty stack.

- Definitions:
  1. A string x is said to be pre-balanced if $L(y) \geq R(y)$ for all prefixes y of x.
  2. A configuration (q, z, α) is said to be blocked if the pda M cannot use up input z, i.e., there is no state r and stack β such that $(q, z, \alpha) \rightarrow^* (r, \varepsilon, \beta)$.
- Facts:
  - 1. If initial configuration $(s, z, \perp)$ is blocked then z is not accepted by M.
  - 2. If (q, z, α) is blocked then (q, zw, α) is blocked for all $w \in \Sigma^*$.

Pf: 1. If $(s, z, \perp)$ is blocked, then there is no state p, stack β such that $(s, z, \perp)$ -->* (p, ε, β), and hence z Is not accepted.

2. Assume (q, zw, α) is not blocked, then there must exists intermediate cfg (p, w, α') such that $(q, zw, \alpha) \rightarrow^* (p, w, \alpha') \rightarrow^* (r, \varepsilon, \beta)$. But $(q, zw, \alpha) \rightarrow^* (p, w, \alpha')$ implies $(q, z, \alpha) \rightarrow^* (p, \varepsilon, \alpha'')$ and (q, z, α) is not blocked.

- Lemma 1: For all strings z,x,
  - if z is prebalanced then $(q, zx, \perp) \to^* (q, x, \alpha\perp)$ iff $\alpha = [^{L(z)-R(z)}$;
  - if z is not prebalanced, $(q, z, \perp)$ is blocked.

Pf: By induction on z.

basic case: $z = \varepsilon$. Then $(q, zx, \perp) = (q, x, \perp) \to^0 (q, x, \alpha\perp)$ iff $\alpha = [^{L(z)-R(z)}$.

inductive case: z = ya, where a is '[' or ']'.

case 1: z = y[.

If y is prebalanced, then so is z. By ind. hyp. $(q, zx, \perp) = (q, y[, \perp) \to^* (q, [x, [^{L(y)-R(y)}\perp) \to (q, x, [[^{L(y)-R(y)}\perp) = (q, x, [^{L(z)-R(z)}\perp)$.

If y is not prebalanced, then, by ind. hyp., $(q, y, \perp)$ is blocked and hence $(q, y[, \perp)$ is blocked as well.

case 2: z = y].

If y is not prebalanced, then neither is z. By ind. hyp. $(q, y, \perp)$ is blocked, hence $(q, y], \perp)$ is blocked

If y is prebalanced and L(y) = R(y). Then z is not prebalanced.

By ind. hyp., if $(q, y], \perp) \to^* (q,], \alpha\perp)$ then $\alpha = [^{L(z)-R(z)} = \varepsilon$, but then $(q,], \perp)$ is blocked. Hence $(q, z, \perp)$ is blocked.

Finally, if y is prebalanced and L(y) > R(y). Then z is prebalanced, and

$(q,y]x,\perp) \text{-->}^* (q,]x, [^{L(y)-R(y)} \perp)$ --- ind. hyp

$\text{-->} (q, x, [^{L(y)-R(y)-1}\perp)$ --- (iii)

$= (q, x, [^{L(z)-R(z)}\perp)$

On the other hand, if

$(q,y]x,\perp) \text{-->}^* (q,x, \alpha\perp)$ .Then there must exist a cfg $(q,]x, \beta)$ such that

$(q,y]x,\perp) \text{-->}^* (q,]x, \beta) \text{-->}^* (q,x, \alpha\perp)$.

But then the intructions executed in the last part must be IV* III IV*.

If $(q,]x, \beta) \text{-->}_{IV^*\ III\ IV^*} (q,x, \alpha\perp)$, then $\beta = \perp^m[\perp^v\alpha\perp$. But by ind. hyp., $\beta = [^{L(y)-R(y)}\perp$, hence m = 0, n = 0 and $\alpha = [^{L(y)-R(y)-1}\perp$.

Pf [of theorem 1] : Let x be any string.

If x is balanced, then it is prebalanced and L(x) – R(x) = 0. Hence, by lemma 1, $(q, x\varepsilon,\perp) \text{-->}^* (q, \varepsilon, [^0\perp) \text{-->}_{IV} (q, \varepsilon, \varepsilon)$. As a result, x is accepted.

If x is not balanced, it is not prebalanced. Hence, by lemma 1, $(q, x,\perp)$ is blocked and is not accepted.

ANOTHER EXAMPLE

- The set $\{ww \mid w \in \{a,b\}^*\}$ is known to be not Context-free but its complement
  $L_1 = \{a,b\}^* - \{ww \mid w \in \{a,b\}^*\}$ is.

Exercise: Design a NPDA to accept L_1 by empty stack.

Hint: $x \in L_1$ iff

(1) $|x|$ is odd or

(2) $x = yazybz'$ or $ybzyaz'$ for some $y,z,z' \in \{a,b\}^*$ with $|z|=|z'|$, which also means
$x = yay'ubu'$ or $yby'uau'$ for some $y,y',u,u' \in \{a,b\}^*$
with $|y|=|y'|$ and $|u|=|u'|$.

EQUIVALENT EXPRESSIVE POWER OF BOTH TYPES OF ACCEPTANCE

- $M = (Q,\Sigma,\Gamma,\delta,s,,F)$: a PDA

  Let u, t : two new states $\notin$ Q and

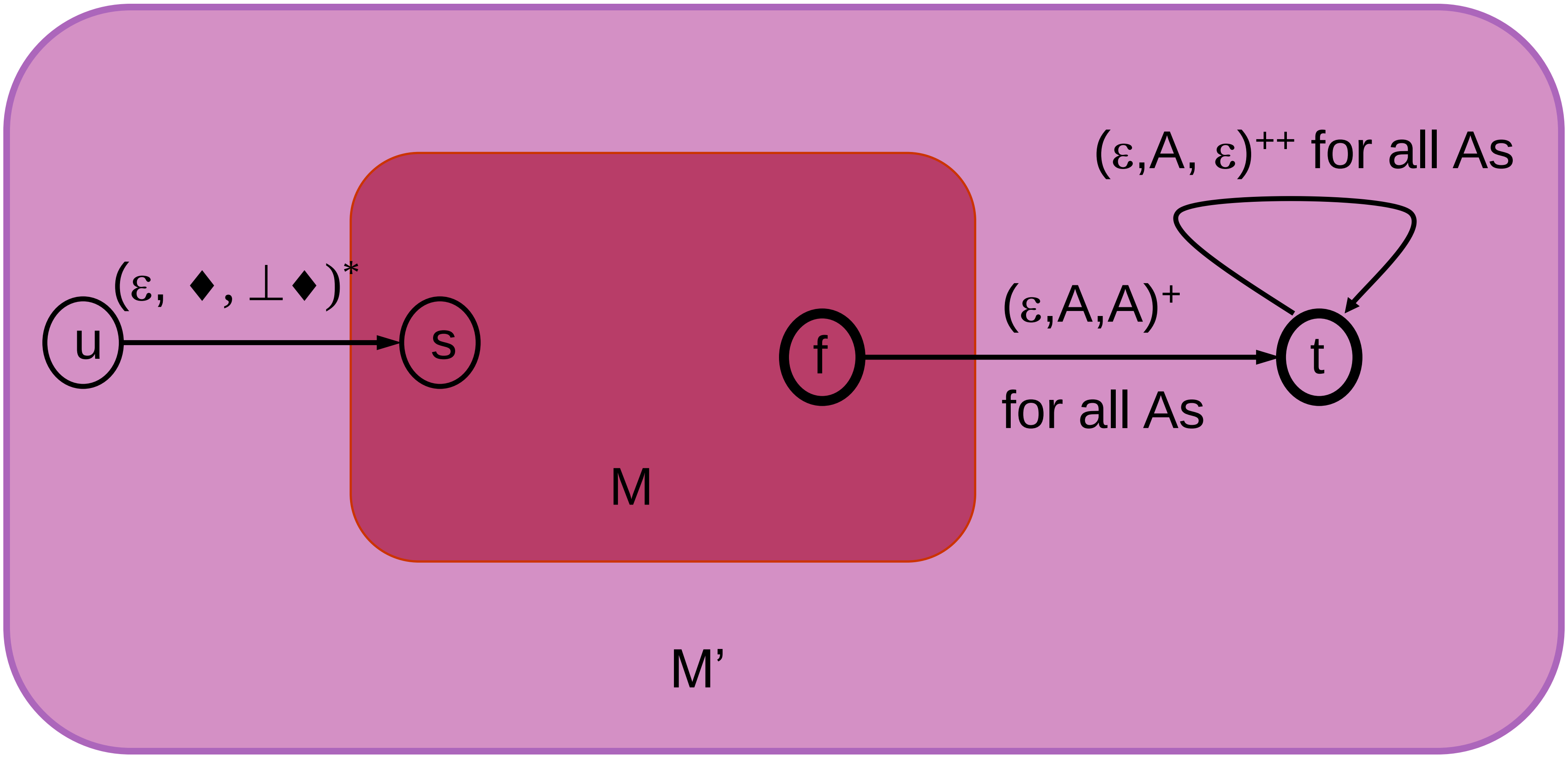
  ♦ : a new stack symbol $\notin \Gamma$.
- Define a new PDA $M' = (Q',\Sigma,\Gamma',\delta',s',$ ♦, $F')$ where
  - $Q' = Q \cup \{u, t\}$, $\Gamma' = \Gamma \cup \{$♦$\}$, $s' = u$, $F' = \{t\}$ and
  - $\delta' = \delta \cup \{ (u,\varepsilon,$ ♦ $) \dashrightarrow (s, \perp$♦$) \}$ // push $\perp$ and call M
  - $\cup \{ (f, \varepsilon, A) \to (t,A) \mid f \in F$ and $A \in \Gamma' \}$ /* return to M'
  - after reaching final states */
  - $\cup \{(t, \varepsilon,A) \dashrightarrow (t,\varepsilon) \mid A \in \Gamma' \}$ // pop until EmptyStack
- Diagram form relating M and M': see next slide.

Theorem: $L_f(M) = L_e(M')$

pf: M accepts x => $(s, x, \perp) \dashrightarrow^n_M (q, \varepsilon, \gamma)$ for some $q \in F$

=> $(u, x,$ ♦ $) \dashrightarrow_{M'} (s, x, \perp$♦$) \dashrightarrow^n_{M'} (q, \varepsilon, \gamma$♦$) \dashrightarrow_{M'} (t, \varepsilon, \gamma$♦$)$

$\dashrightarrow^*_{M'} (t,\varepsilon, \varepsilon)$ => M' accepts x by empty stack.



*: push ⊥ and call M

+: return to t of M' once reaching final states of M

++: pop all stack symbols until emptystack

FROM FINALSTATE TO EMPTYSTACK

Conversely, M' accepts x by empty stack

=> $(u, x, \blacklozenge) \dashrightarrow_{M'} (s, x, \perp\blacklozenge) \dashrightarrow^*_{M'} (q, y, \gamma\blacklozenge) \dashrightarrow (t, y, \gamma\blacklozenge) \dashrightarrow^*$

$(t, \varepsilon, \varepsilon)$ for some $q \in F$

$\Rightarrow$ $y = \varepsilon$ since M' cannot consume any input symbol after it enters state t. => M accepts x by final state.

- Define next new PDA M'' = $(Q', \Sigma, \Gamma', \delta'', s', \blacklozenge, F')$ where
  - $Q' = Q \cup \{u, t\}$, $\Gamma' = \Gamma \cup \{\blacklozenge\}$, $s' = u$, $F' = \{t\}$ and
  - $\delta'' = \delta \cup \{ (u, \varepsilon, \blacklozenge) \dashrightarrow (s, \perp\blacklozenge) \}$ // push $\perp$ and call M
  - $\cup \{ (p, \varepsilon, \blacklozenge) \rightarrow (t, \varepsilon) \mid p \in Q \}$ /* return to M'' and accept
  - if EmptyStack */
  -
- Diagram form relating M and M'': See slide 15.

FROM EMPTYSTACK TO FINALSTATE

- Theorem: $L_e(M) = L_f(M'')$.

pf: M accepts x => (s, x, ⊥) -->^n_M (q, ε , ε)

=> (u, x, ♦) $\text{-->}_{M''}$ (s, x, ⊥♦) $\text{-->}^n_{M''}$ (q, ε , ε ♦) $\text{-->}_{M''}$ (t, ε , ε)

=> M'' accepts x by final state (and empty stack).

Conversely, M'' accepts x by final state (and empty stack)

=> (u, x, ♦) $\text{-->}_{M''}$ (s, x, ⊥♦) $\text{-->}^*_{M''}$ (q, y, ♦) $\text{-->}_{M''}$ (t, ε, ε) for

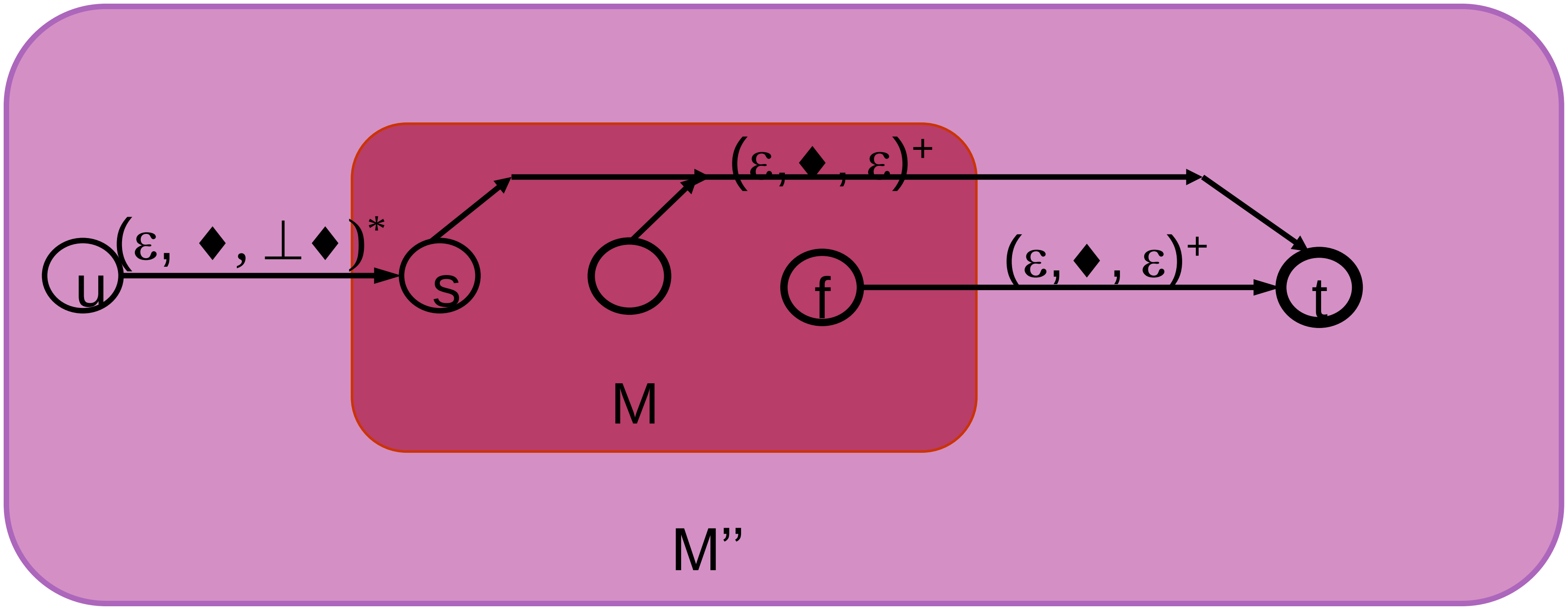
some state q in Q

=> y = ε [and STACK= ε] since M'' does not consume any input symbol at the last transition ((q, ε , ♦), (t, ε))

=> M accepts x by empty stack.

QED

FROM EMPTYSTACK TO FINAL STATE (AND EMPTYSTACK)



* : push ⊥ and call M

+: if emptystack (i.e.see ♦ on stack) ,

then pop ♦ and return to state t of M''

EQUIVALENCE OF PDAS AND CFGS

- Every CFL can be accepted by a PDA.
- G = (N, Σ ,P,S) : a CFG.
  - wlog assume all productions of G are of the form:
  - $A \rightarrow c\ B_1B_2B_3\ldots B_k$ ($k \geq 0$) and $c \in \Sigma \cup \{\varepsilon\}$.
  - note: 1. $A \rightarrow \varepsilon$ satisfies such constraint; 2. can require $k \leq 2$.
- Define a PDA M = ({q}, Σ, N, δ, q, S, {}) from G where
  - q is the only state (hence also the start state),
  - Σ, the set of terminal symbols of G, is the input alphabet of M,
  - N, the set of nonterminals of G, is the stack alphabet of M,
  - S, the start nonterminal of G, is the initial stack symbol of M,
  - {} is the set of final states. (hence M accepts by empty stack!!)
  - $\delta = \{\ ((q,c,A), (q, B_1B_2\ldots B_k))\ \mid A \rightarrow c\ B_1B_2B_3\ldots B_k \in P\ \}$

EXAMPLE

- G : 1. S -> [B S
  2. S -> [B
  3. S-> [S B
  4. S -> [S B S
  5. B ->]

==> δ :

(q, [, S) --> (q, B S)
(q, [, S) --> (q, B)
(q, [, S) --> (q, S B)
(q, [, S) --> (q, S B S)
(q,], B) --> (q, ε)

- L(G) = the set of nonempty balanced parentheses.
- leftmost derivation v.s. computation sequence
  (see next table)

S ^{L}-->$^{*}_{G}$ [[[ ]] [ ]] <==> (q, [[[]][]], S) -->$^{*}_{M}$ (q, ε, ε)

rule applied	sentential form of left-most derivation	configuration of the pda accepting x
	S	(q, [[[ ]] [ ]], S)
3	[S B	(q, [[[[ ]] [ ]], SB)
4	[[S B S B	(q, [[[ ]] [ ]], SBSB)
2	[[[B B S B	(q, [[[ ]] [ ]], BBSB)
5	[[[ ] B S B	(q, [[[ ]] [ ]], BSB)
5	[[[ ]] S B	(q, [[[ ]] [ ]], SB)
2	[[[ ]] [B B	(q, [[[ ]] [ ]], BB)
5	[[[ ]] [ ] B	(q, , [[[ ]] [ ]], B)
5	[[[[ ]] [ ]]	(q, , [[[ ]] [ ]] ,)

LEFTMOST DERIVATION V.S. COMPUTATION SEQUENCE

Lemma 24.1: For any z,y $\in \Sigma^*$, $\gamma \in N^*$ and A $\in$ N,

$A \overset{L}{-->}^n_G z\gamma$ iff $(q, zy, A) -->^n_M (q, y, \gamma)$

Ex: $S \overset{L}{-->}^3_G$ [[[BBSB <==> (q, [[[ ]][]] , S) $-->^3_M$ (q,]][]], BBSB)

pf: By ind. on n.

Basis: n = 0. $A \overset{L}{-->}^0_G z\gamma$ iff $z = \varepsilon$ and $\gamma = A$

iff $(q, zy, A) = (q, y, \gamma)$ iff $(q, zy, A) -->^0_M (q,y,\gamma)$

Ind. case: 1. (only-if part)

Suppose $A \overset{L}{-->}^{n+1}_G z\gamma$ and $B \to c\beta$ was the last rule applied.

I.e., $A \overset{L}{-->}^n_G uB\alpha \overset{L}{-->}_G uc\beta\alpha = z\gamma$ with $z = uc$ and $\gamma = \beta\alpha$.

Hence $(q, u\,cy, A) -->^n_M (q, cy, B\alpha)$ // by ind. hyp.

$-->_M (q, y, \beta\alpha)$ // since $((q,c,B),(q, \beta)) \in \delta$

LEFTMOST DERIVATION V.S. COMPUTATION SEQUENCE (CONT'D)

2. (if-part) Suppose $(q, zy, A) \dashrightarrow^{n+1}_M (q, y, \gamma)$ and $((q,c,B),(q, \beta)) \in \delta$ is the last transition executed. I.e.,

$(q, zy, A) \dashrightarrow^{n}_M (q, cy, B\alpha) \dashrightarrow_M (q, y, \beta\alpha)$ with $\gamma = \beta\alpha$ and $z = uc$ for some u. But then

$A \ {}^L\!\!\dashrightarrow^{n}_G uB\alpha$ // by ind. hyp.,

${}^L\!\!\dashrightarrow uc\ \beta\alpha = z\ \gamma$ // since by def. $B \rightarrow c\ \beta \in P$

Hence $A\ {}^L\!\!\dashrightarrow^{n+1}_G z\ \gamma$ QED

Theorem 24.2: L(G) = L(M).

pf: $x \in L(G)$ iff $S\ {}^L\!\!\dashrightarrow^{*}_G x$

iff $(q, x, S) \dashrightarrow^{*}_M (q, \varepsilon, \varepsilon)$

iff $x \in L(M)$. QED

SIMULATING PDAS BY CFGS

Claim: Every language accepted by a PDA can be generated by a CFG.

- Proved in two steps:
  - 1. Special case : Every PDA with only one state has an equivalent CFG
  - 2. general case: Every PDA has an equivalent CFG.

- Corollary: Every PDA can be minimized to an equivalent PDA with only one state.

pf: M : a PDA with more than one state.

1. apply step 2 to find an equivalent CFG G
2. apply theorem 24.2 on G , we find an equivalent PDA with only one state.

PDA WITH ONLY ONE STATE HAS AN EQUIVALENT CFG.

- $M = (\{s\}, \Sigma, \Gamma, \delta, s, \perp, \{\})$: a PDA with only one state.

  Define a CFG $G = (\Gamma, \Sigma, P, \perp)$ where

  $$P = \{ A \text{ -> } c\beta \mid ((q, c, A), (q, \beta)) \in \delta \}$$

Note: M ==> G is just the inverse of the transformation :

G ==> M defined at slide 16.

Theorem: L(G) = L(M).

Pf: Same as the proof of Lemma 24.1 and Theorem 24.2.

SIMULATING GENERAL PDAS BY CFGS

- How to simulate arbitrary PDA by CFG ?
  - idea: encode all state/stack information in nonterminals !!

Wlog, assume M = $(Q, \Sigma, \Gamma, \delta, s, \perp, \{t\})$ be a PDA with only one final state and M can empty its stack before it enters its final state. (The general pda at slide 15 satisfies such constraint.)

Let $N \subseteq Q \times \Gamma^* \times Q$. Elements of N are written as <pABCq>.

Define a CFG G = (N, Σ, <s$\perp$t>, P) where

P = { <pAr> → c <q B_1 B_2 …B_k r>

| $((p,c,A), (q, B_1B_2\ldots B_k)) \in \delta$, $k \geq 0$, c $\in \Sigma \cup \{\varepsilon\}$, r $\in$ Q }

U Rules for nonterminals <q B_1 B_2 …B_k r>

RULES FOR $\langle Q\ B_1\ B_2 \ldots B_K\ R \rangle$

We want $\langle qB_1 \ldots B_k\ r \rangle$ to simulate the computation process in PDA M:

$(q, \underline{w}y, \underline{B_1B_2 \ldots B_k}\beta) \mid\!- \ldots \mid\!- (r, y, \beta)$ iff $\langle qB_1 \ldots B_k r \rangle \rightarrow^* w$.

Hence: if k = 0. ie., $\langle qB_1B_2 \ldots B_k r \rangle = \langle q\varepsilon r \rangle$, we should have

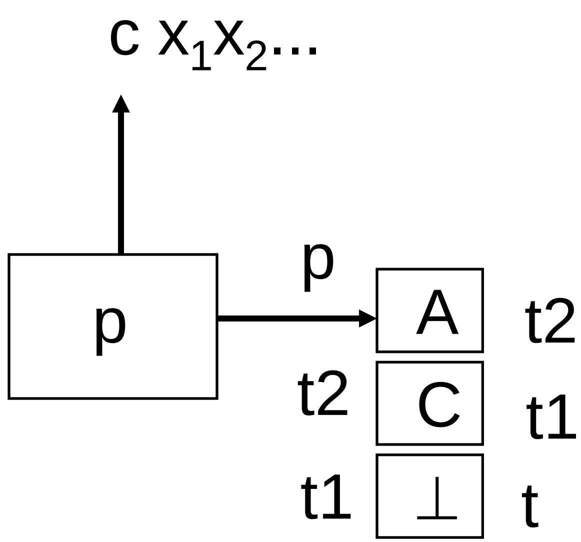
$\langle qr \rangle \rightarrow \varepsilon$ if q = r and

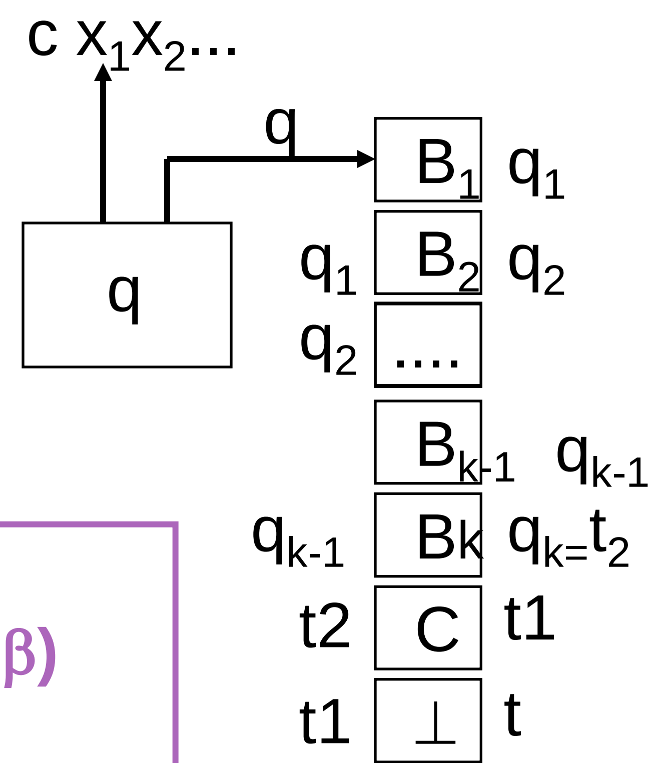
$\langle qr \rangle$ has no rule if $q \neq r$.

If k > 1. Let $B_1B_2 \ldots B_k = B_1\Delta_2$, then :

- $\langle qB_1\Delta_2 r \rangle \rightarrow \Sigma_{u1 \in Q} \langle qB_1u_1 \rangle \langle u_1\Delta_2 r \rangle$
- $\rightarrow \Sigma_{u1 \in Q} \Sigma_{u2 \in Q} \langle qB_1u_1 \rangle \langle u_1B_2u_2 \rangle \langle u_2\Delta_2 r \rangle$
- $\rightarrow \ldots$
- $\rightarrow \Sigma_{u1 \in Q} \Sigma_{u2 \in Q} \ldots \langle qB_1u_1 \rangle \langle u_1B_2u_2 \rangle \ldots \langle u_{k-1}B_kU_k \rangle \langle U_k\Delta_k r \rangle$
- $\rightarrow \Sigma_{u1 \in Q} \Sigma_{u2 \in Q} \ldots \langle qB_1u_1 \rangle \langle u_1B_2u_2 \rangle \ldots \langle u_{k-1}B_k\ r \rangle$

$(p, c, A) \dashrightarrow (q, B_1 B_2 \ldots B_k)$





We want to use $\langle pAq \rangle \rightarrow^* w$ to simulate the computation: $(p, wy, A\beta) \rightarrow^*_M (q, y, \varepsilon\beta)$
So, if $(p,c,A) \rightarrow_M (q, \alpha)$ we have rules :
$\langle p\ A\ r \rangle \rightarrow c \langle q\ \alpha\ r \rangle$ for all states r.

HOW TO DERIVE THE RULE $<P\ A\ R> \rightarrow C\ <Q\ A\ R>$?

How to derive rules for the nonterminal : $<q\ \alpha\ r>$

- case 1: $\alpha = B_1B_2B_3\ldots B_n$ ($n > 0$)
  - => $<q\ \alpha\ r> = <q\ B_1Q\ B_2QB_3Q\ldots QB_nr>$
  - => $<q\ \alpha\ r> \rightarrow <q\ B_1\ q_1> <q_1\ B_2\ q_2> \ldots$
  - $<q_{n-1}\ B_n\ r>$ for all states $q_1, q_2, \ldots, q_{n-1}$ in Q.
- case2: $\alpha = \varepsilon$.
  - $q = r$ => $<q\ \alpha\ r> = <q\ \varepsilon\ r> \rightarrow \varepsilon$.
  - $q \neq r$ => $<q\ \varepsilon\ r>$ cannot derive any string.

  - Then $<pAq> \rightarrow c\ <q\varepsilon q> = c$.

SIMULATING PDAS BY CFG (CONT'D)

- Note: Besides storing sate information on the nonterminals, G simulate M by guessing nondeterministically what states M will enter at certain future points in the computation, saving its guesses on the sentential form, and then verifying later that those guesses are correct.

Lemma 25.1: $(p,x,B_1B_2\ldots B_k) \dashrightarrow^n_M (q,\varepsilon,\varepsilon)$ iff

$\exists\, q_1,q_2,\ldots q_k$ (=q) such that

$$<pB_1q_1><q_1B_2q_2>\ldots<q_{k-1}B_kq> \;{}^{L}\!\!\rightarrow^n_G x. \quad (*)$$

Note: 1. when k = 0 (*) is reduced to $<pq> \;{}^{L}\!\!\rightarrow^n_G x$

2. In particular, $(p,x,B) \dashrightarrow^n_M (q,\varepsilon,\varepsilon)$ iff $<pBq> \;{}^{L}\!\!\rightarrow^n_G x$.

Pf: by ind. on n. Basis: n = 0.

LHS holds iff ($x = \varepsilon$, $k = 0$, and $p = q$) iff RHS holds.

SIMULATING PDAS BY SINGLE-STATE PDAS (CONT'D)

Inductive case:

(=>:) Suppose $(p,x,B_1B_2\ldots B_k) \dashrightarrow^{n+1}{}_M (q,\varepsilon,\varepsilon)$ and

$((p,c,B_1),(r,C_1C_2\ldots C_m))$ is the first instr. executed. I.e.,

$(p,x,B_1B_2\ldots B_k) \dashrightarrow_M (r, y, C_1C_2\ldots C_mB_2\ldots B_k) \dashrightarrow^n{}_M (q,\varepsilon,\varepsilon)$,

where $x = cy$.

By ind. hyp., $\exists$ states $r_1,\ldots,r_{m-1},(r_m = q_1), q_2,\ldots q_{k-1}$ with

$\langle rC_1r_1\rangle\langle r_1C_2r_2\rangle\ldots\langle r_{m-1}C_mq_1\rangle\langle q_1B_2q_2\rangle\ldots\langle q_{k-1}B_kq_k\rangle \;{}^{L}\!\!\rightarrow^n{}_G\; y$

Also by the definition of G:

$\langle pB_1q_1\rangle \rightarrow c\,\langle r_0C_1r_1\rangle\langle r_1C_2r_2\rangle\ldots\langle r_{m-1}C_mq_1\rangle$ is a rule of G.

Combining both, we get:

$\langle pB_1q_1\rangle\,\langle q_1B_2q_2\rangle\ldots\langle q_{k-1}B_kq_k\rangle$

${}^{L}\!\!\rightarrow_G\; c\,\langle r_0C_1r_1\rangle\langle r_1C_2r_2\rangle\ldots\langle r_{m-1}C_mq_1\rangle\,\langle q_1B_2q_2\rangle\ldots\langle q_{k-1}B_kq_k\rangle$

${}^{L}\!\!\rightarrow^n{}_G\; c\,y \quad (= x)$.

SIMULATING PDAS BY CFGS (CONT'D)

(<=:) Suppose $\langle pB_1q_1\rangle\langle q_1\ B_2\ q_2\rangle\ldots\langle q_{k-1}\ B_k\ q\rangle\ {}^{L}\!\rightarrow^{n+1}_{G}\ x$.

Let $\langle pB_1q_1\rangle \rightarrow c\ \langle r_0\ C_1\ r_1\rangle\ \langle r_1\ C_2\ r_2\rangle\ldots\ \langle r_{m-1}\ C_m\ q_1\rangle \in P$ --(*)
be the first rule applied. i.e., Then

$\langle p\ B_1\ q_1\rangle\ \langle q_1\ B_2\ q_2\rangle\ \ldots\langle q_{k-1}\ B_k q\rangle$

${}^{L}\!\rightarrow_G\ c\ \langle r_0\ C_1\ r_1\rangle\ \langle r_1\ C_2\ r_2\rangle\ldots\ \langle r_{m-1}\ C_m\ q_1\rangle\ \langle q_1\ B_2\ q_2\rangle\ \ldots\ \langle q_{k-1}B_kq\rangle$

${}^{L}\!\rightarrow_G^{n}\ cy\ \ (= x\ )$

But then since, by (*), [(p, c, B1) , $(r_0, C_1C_2\ldots C_m)$] – (**) is an instr of M,

$(p,x,B_1\ldots B_k)$ $-->_M$ $(r_0,\ y,\ C_1C_2\ldots C_mB_2\ldots B_n)$ --- By (**)

$-->^n{}_M\ (q,\varepsilon,\varepsilon)$. -- ,by ind. hyp. QED

Theorem 25.2 L(G) = L(M).

Pf: $x \in L(G)$ iff $\langle s\bot t\rangle \rightarrow^* x$

iff $(s,x,\bot)\ -->^*_M\ (t,\varepsilon,\varepsilon)$ ---- Lemma 25.1

iff $x \in L(M)$. QED

EXAMPLE

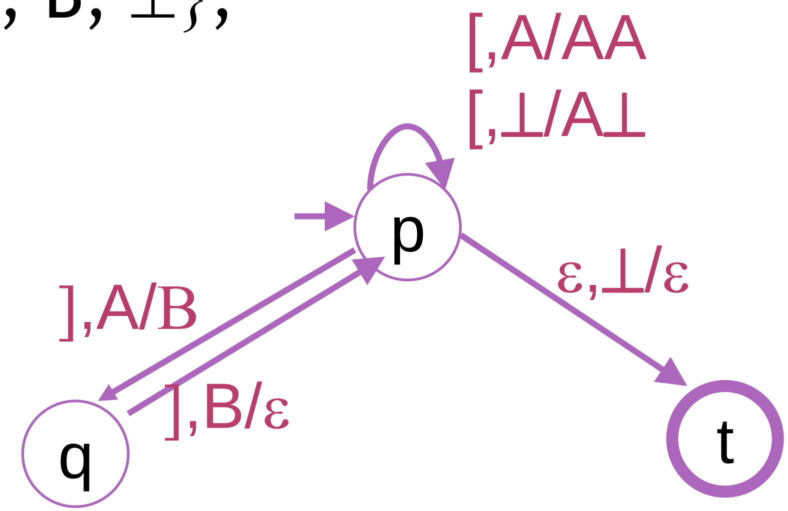
- L = {x∈ {[,]}* | x is a balanced string of [and]], i.e., #](x) = 2 #[(x) and all “]]“s must occur in pairs }
- Ex: [ ]] [[ ]]]] ∈ L but [ ] [ ]]] ∉ L.
- L can be accepted by the PDA

M = (Q, Σ, Γ, δ, p, ⊥,{t}), where

Q = {p,q,t}, Σ = {[,]}, Γ = {A, B, ⊥},

and δ is given as follows:

- (p, [, ⊥) --> (p, A⊥),
- (p,[,A) --> (p,AA),
- (p,], A) --> (q, ε),
- (q,], B) --> (p, ε),
- (p,ε, ⊥) --> (t,ε)



- M can be simulated by the CFG G = (N,Σ, <p⊥t>, P) where
  - N = { <X D Y> | X,Y ∈{p,q,t} and D ∈ { A,B, ⊥ } },
  - and P is derived from the following pseudo rules :
  - (p, [, ⊥) --> (p, A⊥) : <p ⊥ ?> → [<pA⊥?>
  - (p,[,A) --> (p,AA) : <p A $?_1$> → [<pA$?_2$A$?_1$>
  - (p,], A) --> (q, B), : <p A ?> →] <qB?>
  - This produce 3 rules (? = p or q or t).
  - (q,], B) --> (p, ε), : <q B ?> →] <p ε ?>
  - This produces 1 rule :
  - (? = p, but could not be q or t why ?)
  - <q B ?> →] <p ε ?> => <qBp> →] <pεp> $\rightarrow^0$]
  - (p,ε, ⊥) --> (t,ε) : <p⊥?> → <t ε ?>
  - This results in <p⊥t> → ε (since <t ε t> → ε.)

- <p ⊥ ?> → [<pA⊥?> ➔ resulting in 3 rules : ? = p, q or t.
- <p ⊥ p> → [<pA⊥p> ---(1)
- <p ⊥ q> → [<pA⊥q> ---(2)
- <p ⊥ t> → [<pA⊥t> ---(3)
- (1)~(3) each again need to be expanded into 3 rules.
- <pA⊥p> → <pA?><? ⊥ p> where ? is p or q or t.
- <pA⊥q> → <pA?><? ⊥ q> where ? is p or q or t.
- <pA⊥ t> → <pA?><? ⊥ t> where ? is p or q or t.
- <p A $?_1$> → [<pA$?_2$A$?_1$> resulting in 9 rules:
- Where $?_2$ = p,q, or t.
- <p A p> → [<pA$?_2$> <$?_2$⊥p> ---(1)
- <p A q> → [<pA$?_2$> <$?_2$⊥q> ---(2)
- <p A t> → [<pA$?_2$> <$?_2$⊥t> ---(3)