$$\begin{array}{rcl} S & \rightarrow & L = R \mid R \\ L & \rightarrow & *R \mid \mathbf{id} \\ R & \rightarrow & L \end{array} \tag{4.49}$$

$$
\begin{array}{ll}
I_0: & S' \to \cdot S \\
& S \to \cdot L = R \\
& S \to \cdot R \\
& L \to \cdot * R \\
& L \to \cdot \mathbf{id} \\
& R \to \cdot L
\end{array}
$$

$$
\begin{array}{ll}
I_1: & S' \to S \cdot
\end{array}
$$

$$
\begin{array}{ll}
I_2: & S \to L \cdot = R \\
& R \to L \cdot
\end{array}
$$

$$
\begin{array}{ll}
I_3: & S \to R \cdot
\end{array}
$$

$$
\begin{array}{ll}
I_4: & L \to * \cdot R \\
& R \to \cdot L \\
& L \to \cdot * R \\
& L \to \cdot \mathbf{id}
\end{array}
$$

$$
\begin{array}{ll}
I_5: & L \to \mathbf{id} \cdot
\end{array}
$$

$$
\begin{array}{ll}
I_6: & S \to L = \cdot R \\
& R \to \cdot L \\
& L \to \cdot * R \\
& L \to \cdot \mathbf{id}
\end{array}
$$

$$
\begin{array}{ll}
I_7: & L \to * R \cdot
\end{array}
$$

$$
\begin{array}{ll}
I_8: & R \to L \cdot
\end{array}
$$

$$
\begin{array}{ll}
I_9: & S \to L = R \cdot
\end{array}
$$

Figure 4.39: Canonical LR(0) collection for grammar (4.49)

Example 4.51: Let us reconsider Example 4.48, where in state 2 we had item $R \to L\cdot$, which could correspond to $A \to \alpha$ above, and a could be the = sign, which is in FOLLOW(R). Thus, the SLR parser calls for reduction by $R \to L$ in state 2 with = as the next input (the shift action is also called for, because of item $S \to L\cdot{=}R$ in state 2). However, there is no right-sentential form of the grammar in Example 4.48 that begins $R = \cdots$. Thus state 2, which is the state corresponding to viable prefix L only, should not really call for reduction of that L to R. □

It is possible to carry more information in the state that will allow us to rule out some of these invalid reductions by $A \to \alpha$. By splitting states when necessary, we can arrange to have each state of an LR parser indicate exactly which input symbols can follow a handle α for which there is a possible reduction to A.

The extra information is incorporated into the state by redefining items to include a terminal symbol as a second component. The general form of an item becomes $[A \to \alpha \cdot \beta, a]$, where $A \to \alpha\beta$ is a production and a is a terminal or the right endmarker \$. We call such an object an *LR(1) item*. The 1 refers to the length of the second component, called the *lookahead* of the item.[6] The lookahead has no effect in an item of the form $[A \to \alpha{\cdot}\beta, a]$, where β is not ϵ, but an item of the form $[A \to \alpha\cdot, a]$ calls for a reduction by $A \to \alpha$ only if the next input symbol is a. Thus, we are compelled to reduce by $A \to \alpha$ only on those input symbols a for which $[A \to \alpha\cdot, a]$ is an LR(1) item in the state on top of the stack. The set of such a's will always be a subset of FOLLOW(A), but it could be a proper subset, as in Example 4.51.

Example 4.54: Consider the following augmented grammar.

$$\begin{array}{lcl} S' & \rightarrow & S \\ S & \rightarrow & C\,C \\ C & \rightarrow & c\,C \mid d \end{array} \tag{4.55}$$

We begin by computing the closure of $\{[S' \rightarrow \cdot S, \$]\}$. To close, we match the item $[S' \rightarrow \cdot S, \$]$ with the item $[A \rightarrow \alpha \cdot B\beta, a]$ in the procedure CLOSURE. That is, $A = S'$, $\alpha = \epsilon$, $B = S$, $\beta = \epsilon$, and $a = \$$. Function CLOSURE tells us to add $[B \rightarrow \cdot\gamma, b]$ for each production $B \rightarrow \gamma$ and terminal b in FIRST(βa). In terms of the present grammar, $B \rightarrow \gamma$ must be $S \rightarrow CC$, and since β is ϵ and a is \$, b may only be \$. Thus we add $[S \rightarrow \cdot CC, \$]$.

We continue to compute the closure by adding all items $[C \rightarrow \cdot\gamma, b]$ for b in FIRST$(C\$)$. That is, matching $[S \rightarrow \cdot CC, \$]$ against $[A \rightarrow \alpha \cdot B\beta, a]$, we have $A = S$, $\alpha = \epsilon$, $B = C$, $\beta = C$, and $a = \$$. Since C does not derive the empty string, FIRST$(C\$)$ = FIRST(C). Since FIRST(C) contains terminals c and d, we add items $[C \rightarrow \cdot cC, c]$, $[C \rightarrow \cdot cC, d]$, $[C \rightarrow \cdot d, c]$ and $[C \rightarrow \cdot d, d]$. None of the new items has a nonterminal immediately to the right of the dot, so we have completed our first set of LR(1) items. The initial set of items is

$$\begin{array}{ll} I_0: & S \rightarrow \cdot S, \$ \\ & S \rightarrow \cdot CC,\ \$ \\ & C \rightarrow \cdot cC,\ c/d \\ & C \rightarrow \cdot d,\ c/d \end{array}$$

The brackets have been omitted for notational convenience, and we use the notation $[C \to \cdot cC,\ c/d]$ as a shorthand for the two items $[C \to \cdot cC,\ c]$ and $[C \to \cdot cC,\ d]$.

Now we compute $\text{GOTO}(I_0, X)$ for the various values of X. For $X = S$ we must close the item $[S' \to S\cdot,\ \$]$. No additional closure is possible, since the dot is at the right end. Thus we have the next set of items

$$I_1 : \quad S' \to S\cdot,\ \$$$

For $X = C$ we close $[S \to C{\cdot}C,\ \$]$. We add the C-productions with second component \$ and then can add no more, yielding

$$\begin{aligned} I_2 : \quad & S \to C{\cdot}C,\ \$ \\ & C \to \cdot cC,\ \$ \\ & C \to \cdot d,\ \$ \end{aligned}$$

Next, let $X = c$. We must close $\{[C \to c{\cdot}C,\ c/d]\}$. We add the C-productions with second component c/d, yielding

$$
\begin{aligned}
I_3: \quad & C \to c{\cdot}C,\ c/d \\
& C \to {\cdot}cC,\ c/d \\
& C \to {\cdot}d,\ c/d
\end{aligned}
$$

Finally, let $X = d$, and we wind up with the set of items

$$
I_4: \quad C \to d{\cdot},\ c/d
$$

We have finished considering GOTO on I_0. We get no new sets from I_1, but I_2 has goto's on C, c, and d. For GOTO$(I_2,\ C)$ we get

$$
I_5: \quad S \to CC{\cdot}, \$
$$

no closure being needed. To compute GOTO$(I_2,\ c)$ we take the closure of $\{[C \to c{\cdot}C,\ \$]\}$, to obtain

$$
\begin{aligned}
I_6: \quad & C \to c{\cdot}C,\ \$ \\
& C \to {\cdot}cC,\ \$ \\
& C \to {\cdot}d,\ \$
\end{aligned}
$$

Note that I_6 differs from I_3 only in second components. We shall see that it is common for several sets of LR(1) items for a grammar to have the same first components and differ in their second components. When we construct the collection of sets of LR(0) items for the same grammar, each set of LR(0) items will coincide with the set of first components of one or more sets of LR(1) items. We shall have more to say about this phenomenon when we discuss LALR parsing.

Continuing with the GOTO function for I_2, GOTO(I_2, d) is seen to be

$$I_7: \quad C \rightarrow d\cdot,\ \$$$

Turning now to I_3, the GOTO's of I_3 on c and d are I_3 and I_4, respectively, and GOTO(I_3, C) is

$$I_8: \quad C \rightarrow cC\cdot,\ c/d$$

I_4 and I_5 have no GOTO's, since all items have their dots at the right end. The GOTO's of I_6 on c and d are I_6 and I_7, respectively, and GOTO(I_6, C) is

$$I_9: \quad C \rightarrow cC\cdot,\ \$$$

The remaining sets of items yield no GOTO's, so we are done. Figure 4.41 shows the ten sets of items with their goto's. □

To appreciate the new definition of the CLOSURE operation, in particular, why b must be in FIRST(βa), consider an item of the form $[A \rightarrow \alpha \cdot B\beta, a]$ in the set of items valid for some viable prefix γ. Then there is a rightmost derivation $S \underset{rm}{\overset{*}{\Rightarrow}} \delta A a x \underset{rm}{\Rightarrow} \delta\alpha B\beta a x$, where $\gamma = \delta\alpha$. Suppose $\beta a x$ derives terminal string by. Then for each production of the form $B \rightarrow \eta$ for some η, we have derivation $S \underset{rm}{\overset{*}{\Rightarrow}} \gamma B b y \underset{rm}{\Rightarrow} \gamma\eta b y$. Thus, $[B \rightarrow \cdot\eta, b]$ is valid for γ. Note that b can be the first terminal derived from β, or it is possible that β derives ϵ in the derivation $\beta a x \underset{rm}{\overset{*}{\Rightarrow}} by$, and b can therefore be a. To summarize both possibilities we say that b can be any terminal in FIRST$(\beta a x)$, where FIRST is the function from Section 4.4. Note that x cannot contain the first terminal of by, so FIRST$(\beta a x)$ = FIRST(βa). We now give the LR(1) sets of items construction.

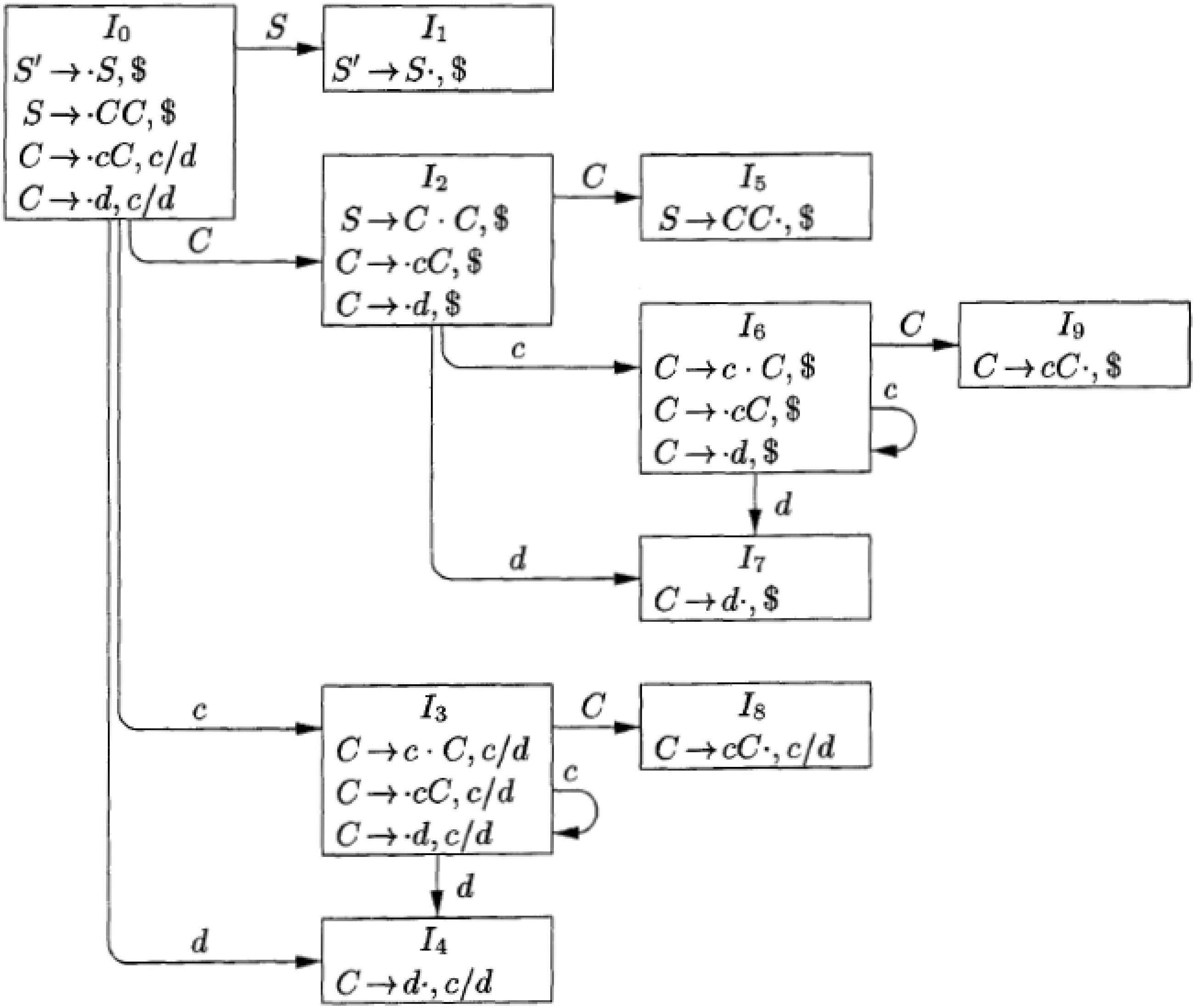

Figure 4.41: The GOTO graph for grammar (4.55)

Algorithm 4.56: Construction of canonical-LR parsing tables.

INPUT: An augmented grammar G'.

OUTPUT: The canonical-LR parsing table functions ACTION and GOTO for G'.

METHOD:

1. Construct $C' = \{I_0, I_1, \cdots, I_n\}$, the collection of sets of LR(1) items for G'.

2. State i of the parser is constructed from I_i. The parsing action for state i is determined as follows.

   (a) If $[A \to \alpha \cdot a\beta, b]$ is in I_i and $\text{GOTO}(I_i, a) = I_j$, then set $\text{ACTION}[i, a]$ to "shift j." Here a must be a terminal.

   (b) If $[A \to \alpha\cdot, a]$ is in I_i, $A \neq S'$, then set $\text{ACTION}[i, a]$ to "reduce $A \to \alpha$."

   (c) If $[S' \to S\cdot, \$]$ is in I_i, then set $\text{ACTION}[i, \$]$ to "accept."

   If any conflicting actions result from the above rules, we say the grammar is not LR(1). The algorithm fails to produce a parser in this case.

3. The goto transitions for state i are constructed for all nonterminals A using the rule: If $\text{GOTO}(I_i, A) = I_j$, then $\text{GOTO}[i, A] = j$.

4. All entries not defined by rules (2) and (3) are made "error."

5. The initial state of the parser is the one constructed from the set of items containing $[S' \to \cdot S, \$]$.

STATE	ACTION			GOTO	
	c	d	\$	S	C
0	s3	s4		1	2
1			acc		
2	s6	s7			5
3	s3	s4			8
4	r3	r3			
5			r1		
6	s6	s7			9
7			r3		
8	r2	r2			
9			r2		

Figure 4.42: Canonical parsing table for grammar (4.55)

Example 1)

Given the following grammar:

1. $S \rightarrow A$
2. $S \rightarrow xb$
3. $A \rightarrow aAb$
4. $A \rightarrow B$
5. $B \rightarrow b$

Compute the LR(1) items & the corresponding DFA, also construct the parsing table.

Augment the grammar & find First Set:

$S' \rightarrow S$	$First(S') = \{x, a, b\}$
$S \rightarrow A \mid xb$	$First(S) = \{x, a, b\}$
$A \rightarrow aAb \mid B$	$First(A) = \{a, b\}$
$B \rightarrow b$	$First(B) = \{b\}$

0. $S' \rightarrow S$
1. $S \rightarrow A$
2. $S \rightarrow xb$
3. $A \rightarrow aAb$
4. $A \rightarrow B$
5. $B \rightarrow b$

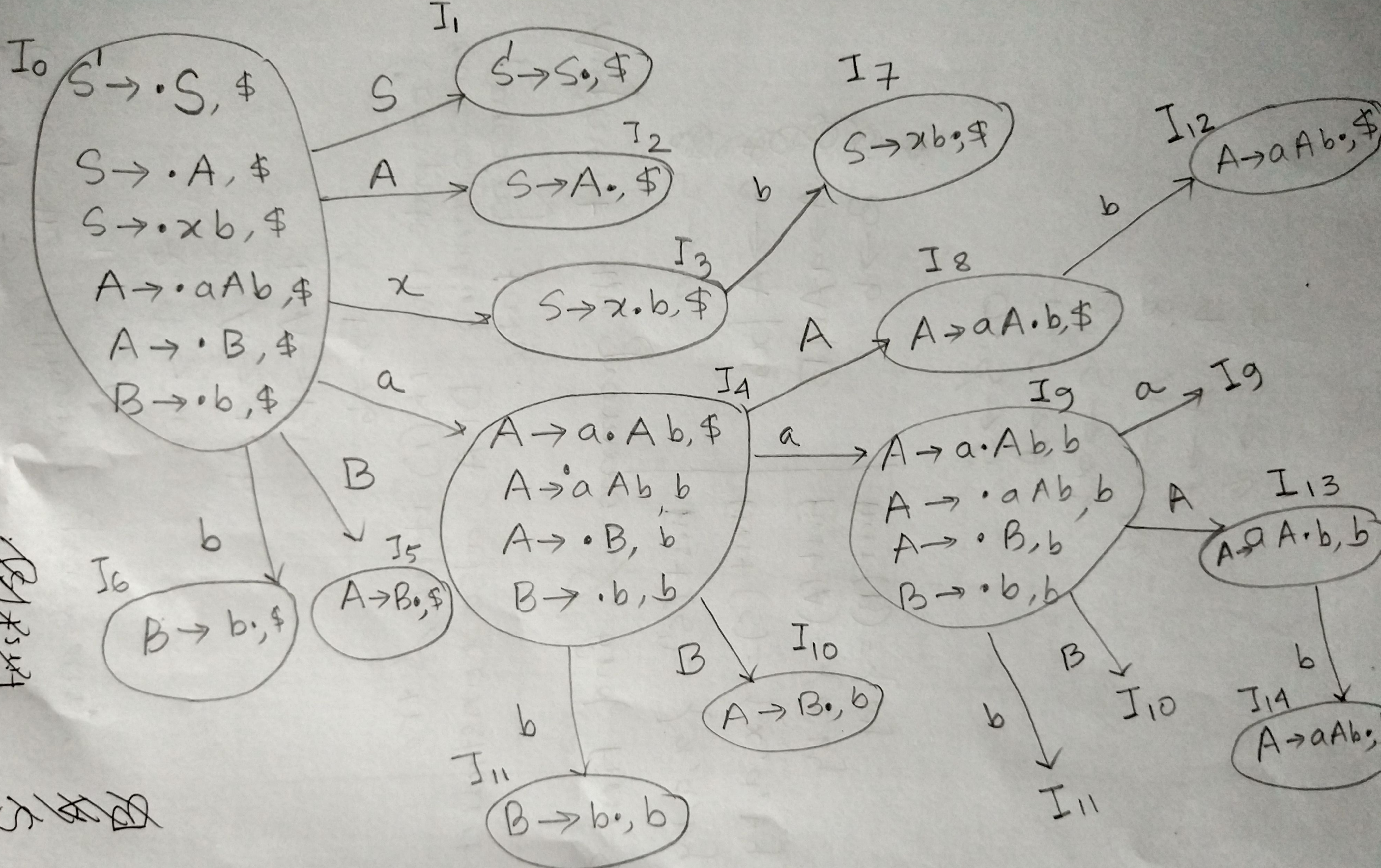

Fig: LR(1) automation

	ACTION (i,a)				GOTO (i,A)		
State	x	a	b	$	S	A	B
0	S3	S4	S6		1	2	5
1				accept			
2				R1			
3			S7				
4		S9	S11			8	10
5				R4			
6				R5			
7				R2			
8			S12				
9		S9	S11			13	10
10			R4				
11			R5				
12				R3			
13			S14				
14			R3				