Building LALR(1) CFSM

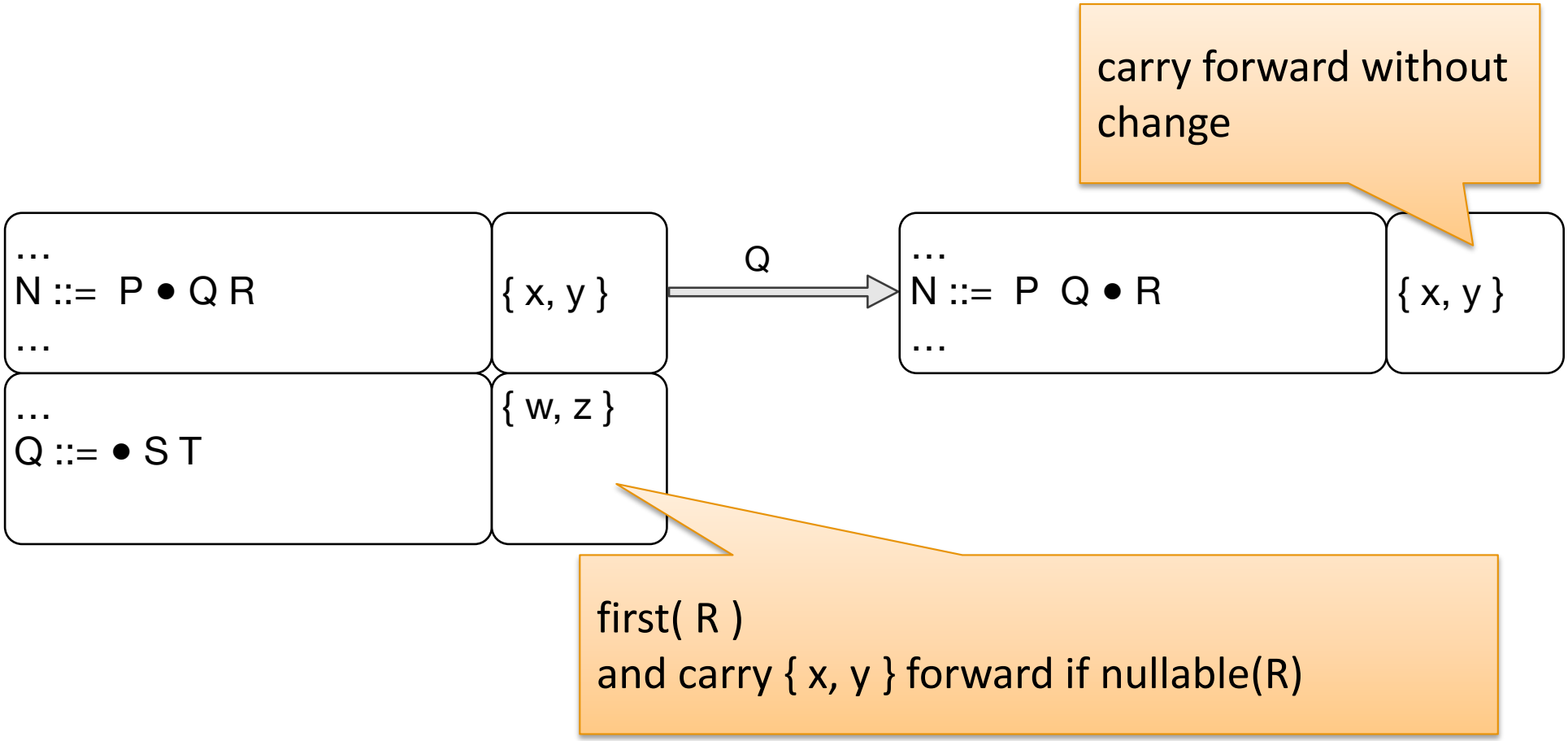
(with lookaheads)

Rules

… N ::= P • Q R …	{ x, y }

Cursor is at non-terminal Q, so …

Expand and propagate



An expression grammar

S ::= E $
E ::= E + T E ::= T
T ::= T * F T ::= F
F ::= i

In layers to express precedence, which will be captured in LALR(1) lookahead.

Build initial state

Grammar

S ::= E $
E ::= E + T E ::= T
T ::= T * F T ::= F
F ::= i

CFSM

item	lookahead
S ::= •E $	{ $ }

Lookahead for initial item is by convention either empty or $; final reduce ("accept") is special

Expand E, with lookahead

Grammar

S ::= E $
E ::= E + T E ::= T
T ::= T * F T ::= F
F ::= i

CFSM

item	lookahead
S ::= •E **$**	{ $ }
E ::= • E + T E ::= • T	{ **$** }

We are expanding an E followed by a $ (eof)

Expand E again

Grammar

S ::= E $
E ::= E + T E ::= T
T ::= T * F T ::= F
F ::= i

CFSM

item	lookahead
S ::= •E $	{ $ }
E ::= • E + T E ::= • T	{ $ + }

We are expanding an E followed by a +

Expand T

Grammar

S ::= E $
E ::= E + T E ::= T
T ::= T * F T ::= F
F ::= i

CFSM

item	lookahead
S ::= •E **$**	{ $ }
E ::= • E **+** T E ::= • T	{ **$** **+** }
T ::= • T * F T ::= • F	{ **$** **+** }

The T is at the end of the production, so carry the lookahead forward

Expand T again

Grammar

S ::= E $
E ::= E + T E ::= T
T ::= T * F T ::= F
F ::= i

CFSM

item	lookahead
S ::= •E **$**	{ $ }
E ::= • E **+** T E ::= • T	{ **$ +** }
T ::= • T * F T ::= • F	{ **$ +** * }

*Now T is followed by **

Expand F

Grammar

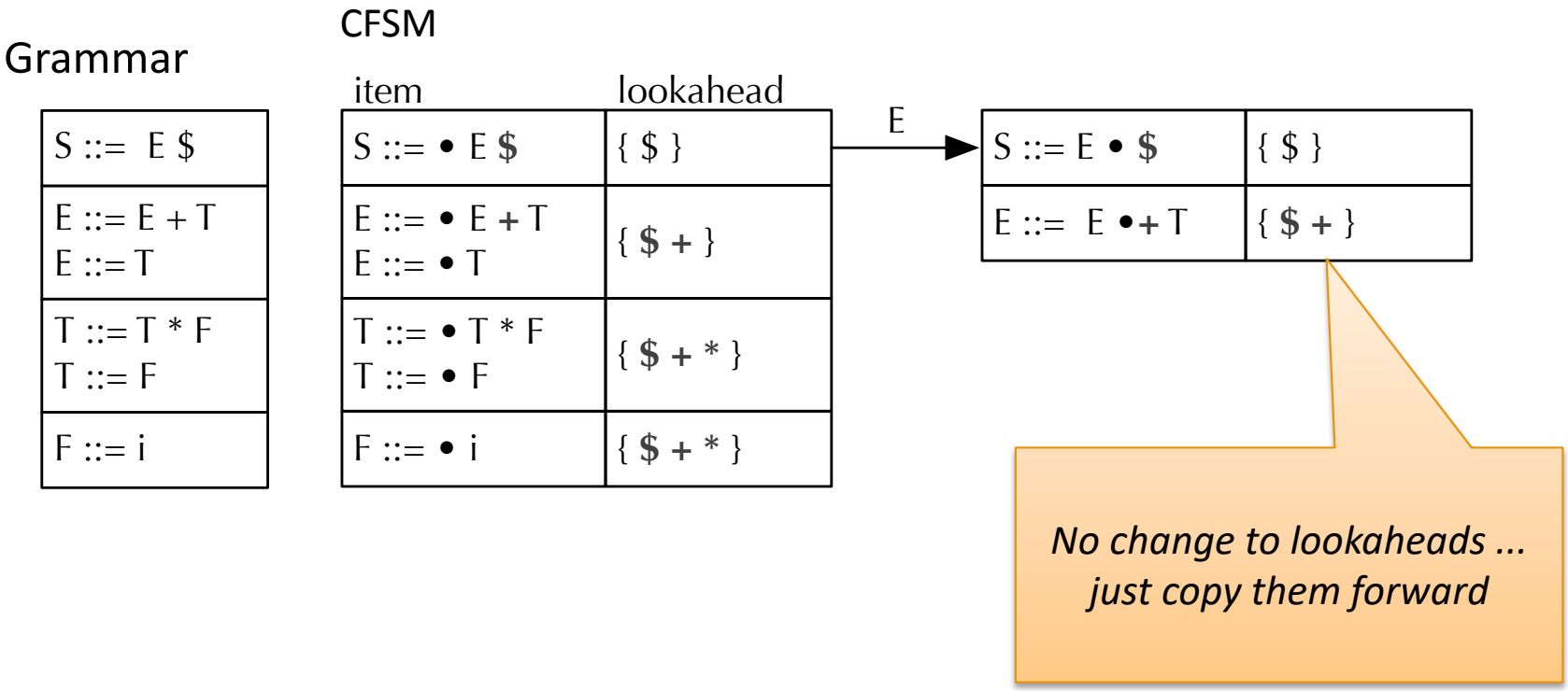
S ::= E $
E ::= E + T E ::= T
T ::= T * F T ::= F
F ::= i

CFSM

item	lookahead
S ::= •E **$**	{ $ }
E ::= • E **+** T E ::= • T	{ **$ +** }
T ::= • T * F T ::= • F	{ **$ +** * }
F ::= • i	{ **$ +** * }

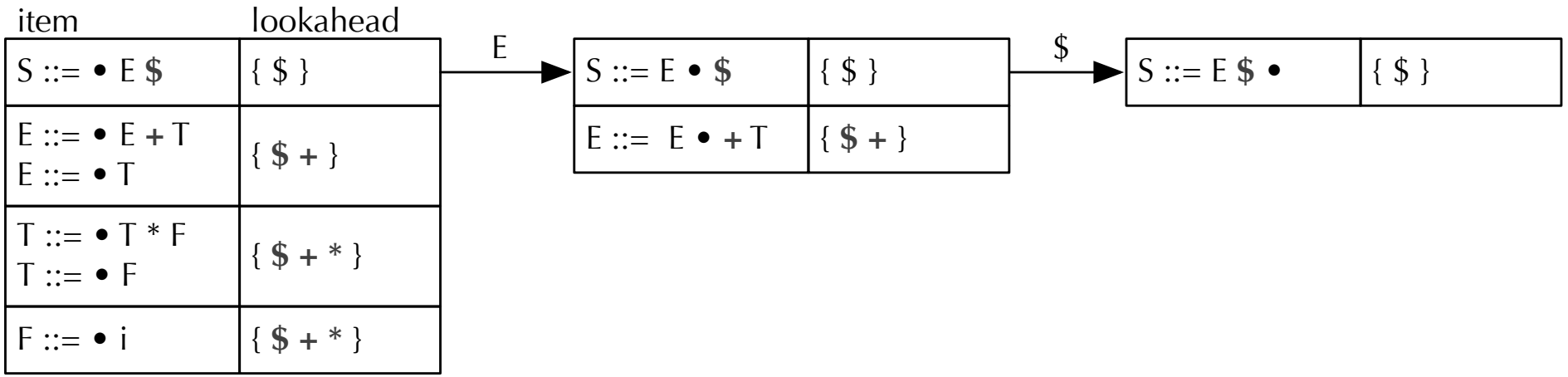
Expanding F at end of production, so carry forward lookahead

"Goto" transition on E ...



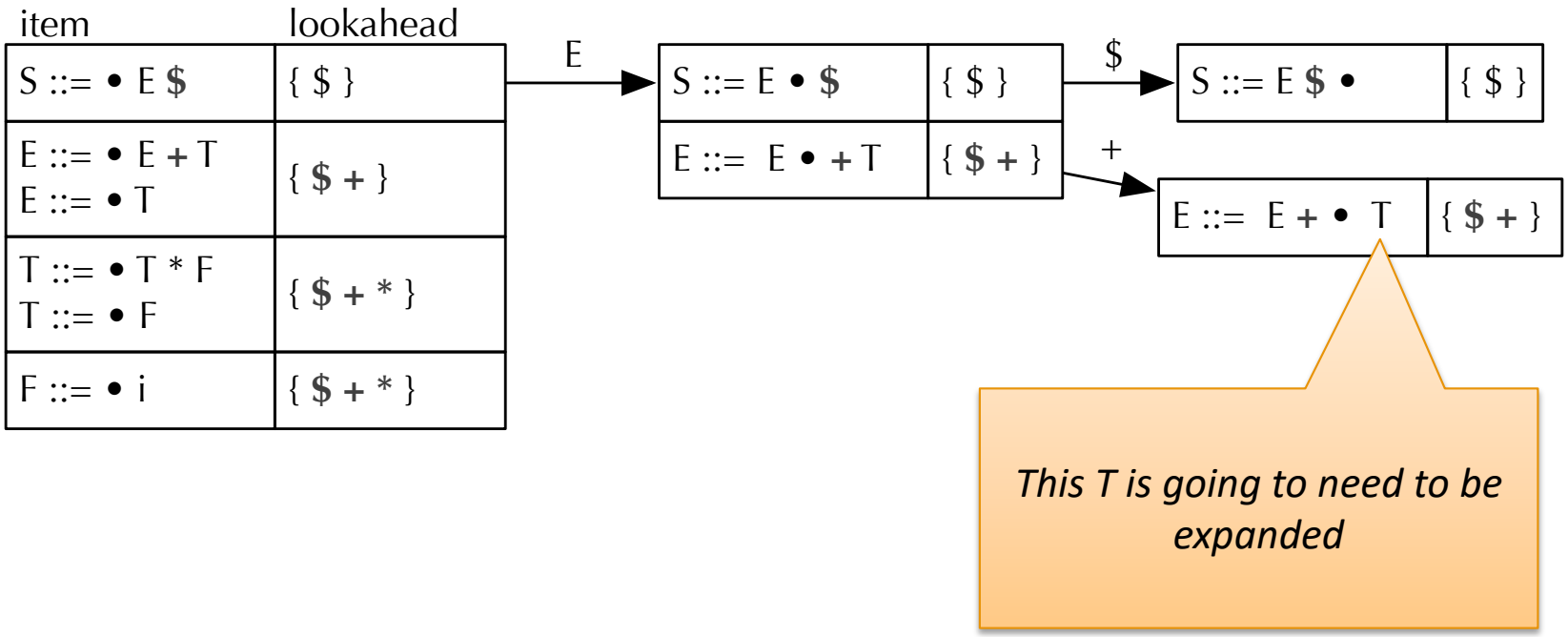
"Accept" transition ("Goto" at EOF)

CFSM



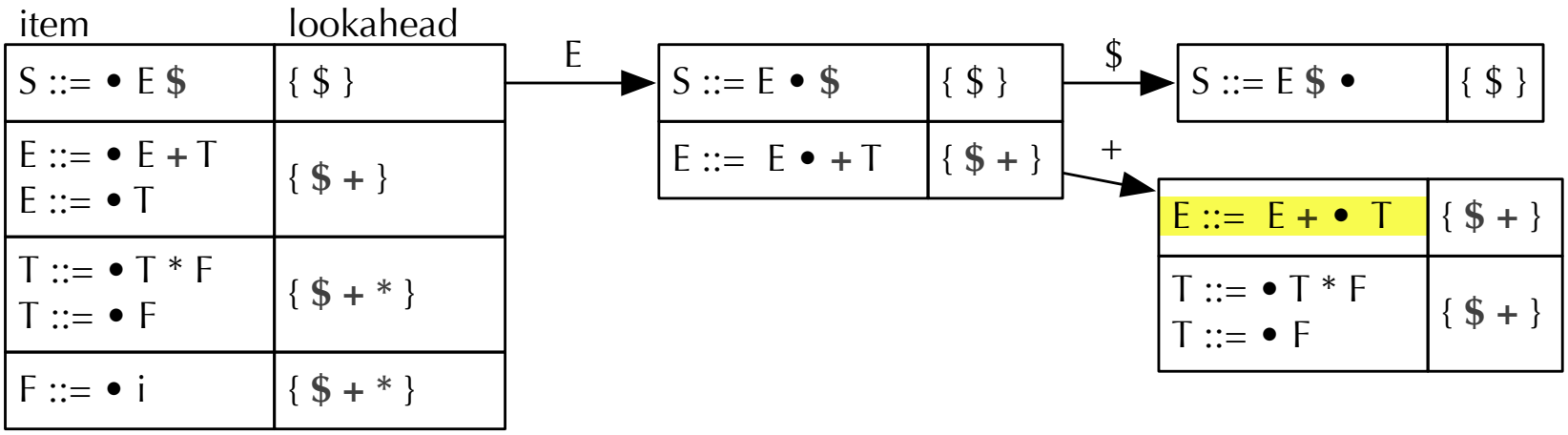
Shift "+"

CFSM


item
lookahead
S ::= • E $
{ $ }
E ::= • E + T
E ::= • T
{ $ + }
T ::= • T * F
T ::= • F
{ $ + * }
F ::= • i
{ $ + * }
E
S ::= E • $
{ $ }
E ::= E • + T
{ $ + }
$
S ::= E $ •
{ $ }
+
E ::= E + • T
{ $ + }
This T is going to need to be expanded

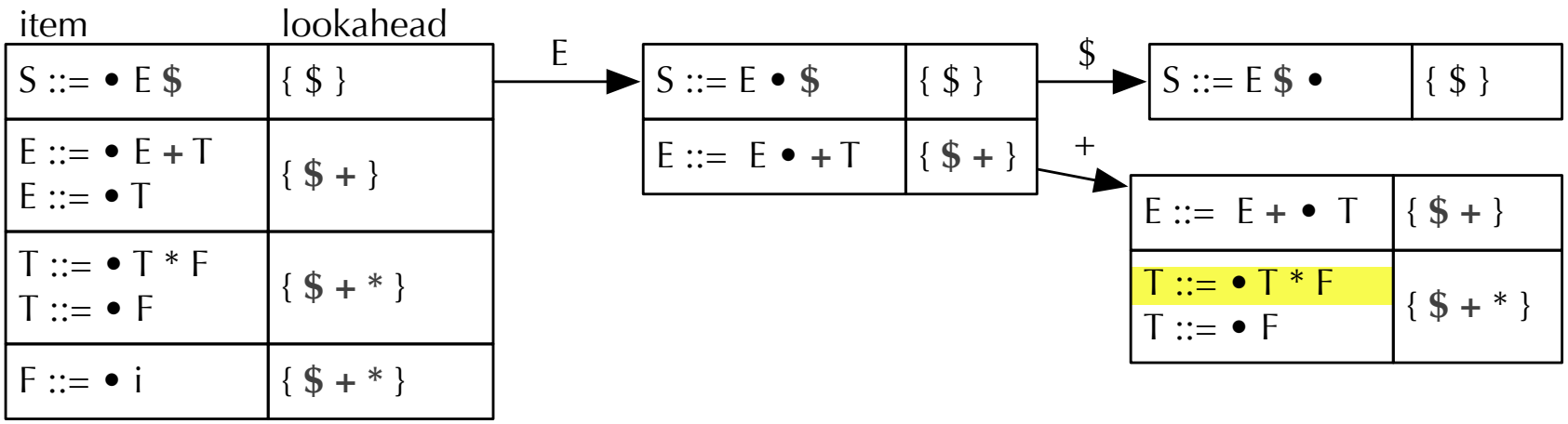
Expand the T …

CFSM



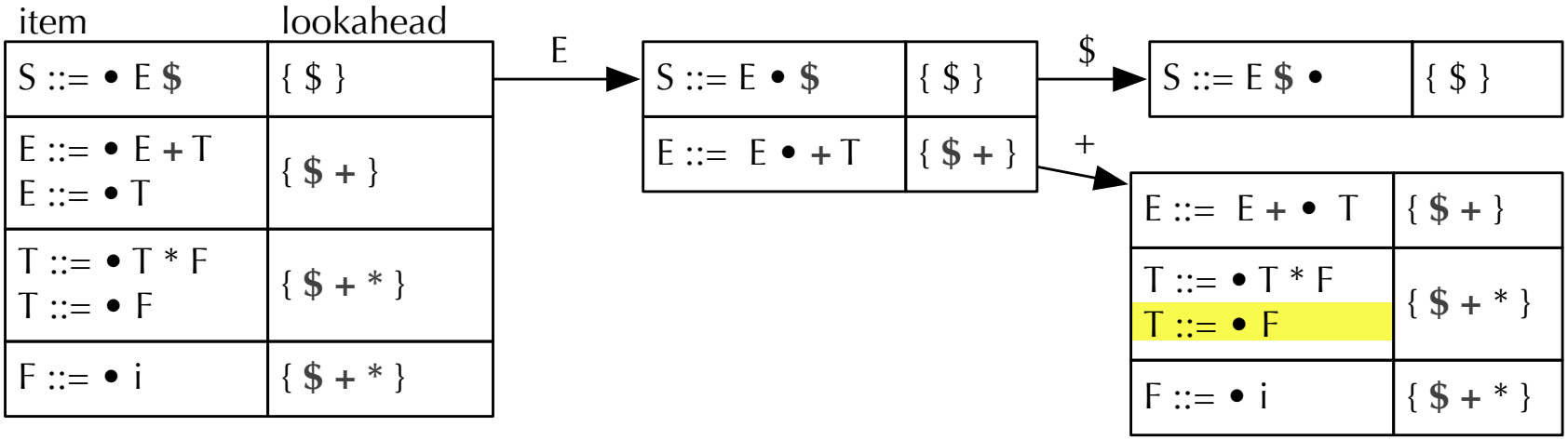
Expand the other T ...

CFSM



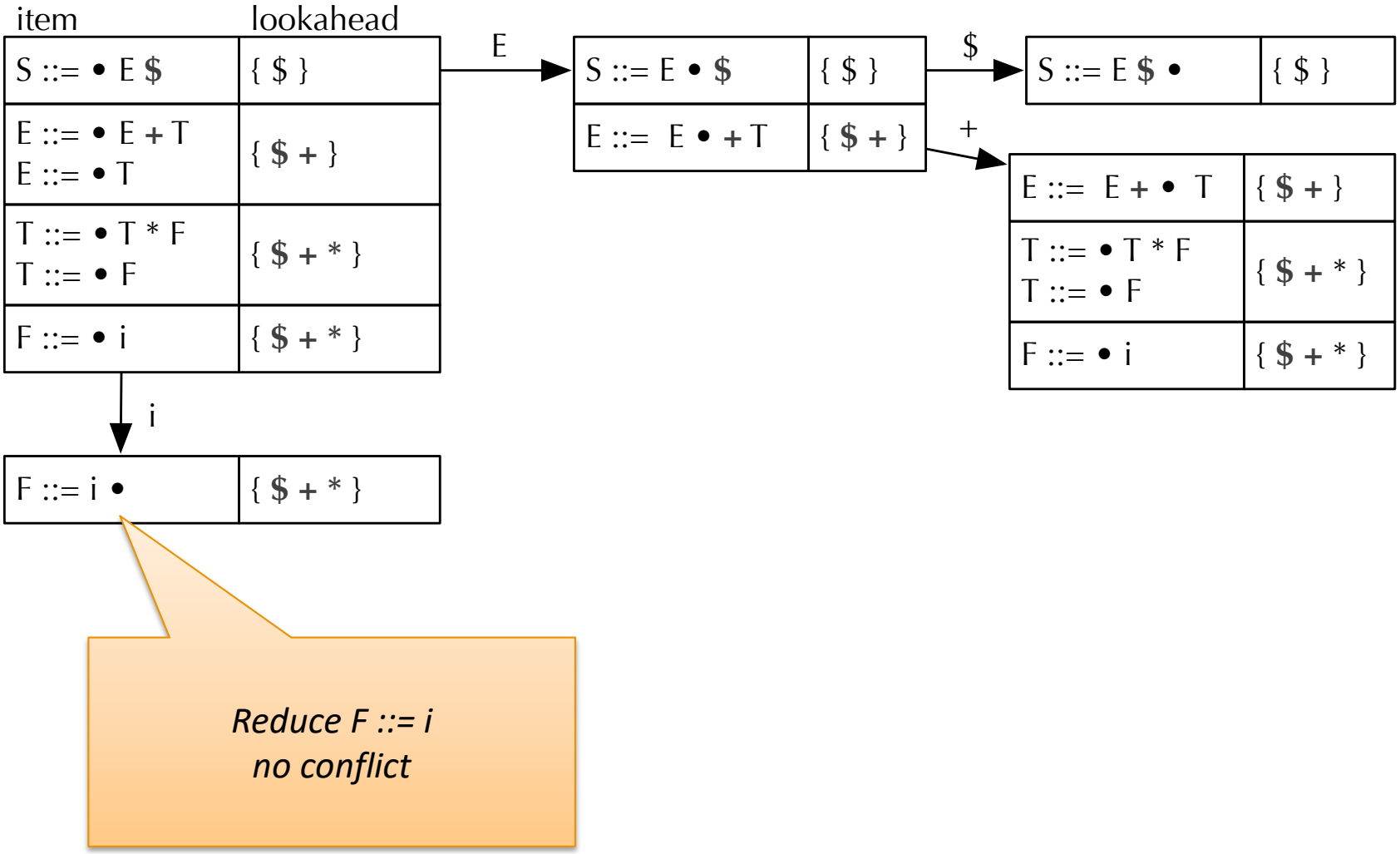
Expand the F

CFSM



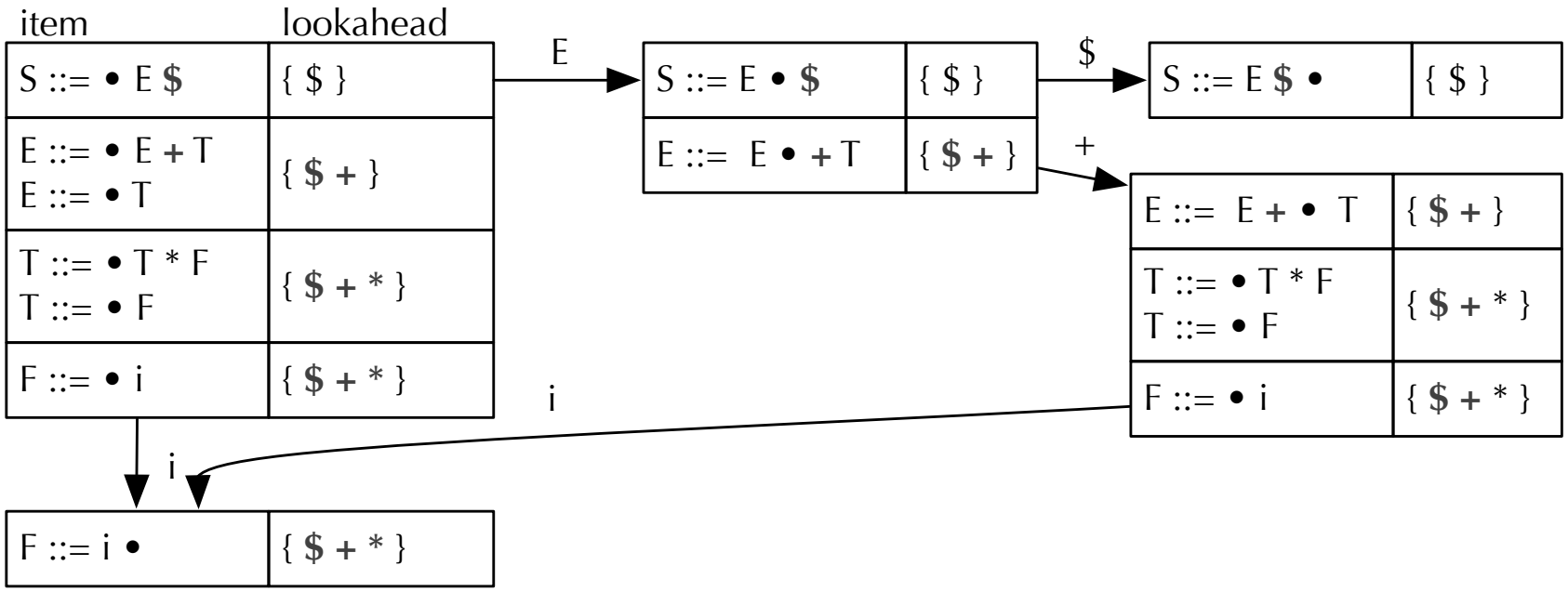
Shift i

CFSM


item
lookahead
S ::= • E $
{ $ }
E ::= • E + T
E ::= • T
{ $ + }
T ::= • T * F
T ::= • F
{ $ + * }
F ::= • i
{ $ + * }
E
S ::= E • $
{ $ }
E ::= E • + T
{ $ + }
$
S ::= E $ •
{ $ }
+
E ::= E + • T
{ $ + }
T ::= • T * F
T ::= • F
{ $ + * }
F ::= • i
{ $ + * }
i
F ::= i •
{ $ + * }
Reduce F ::= i
no conflict

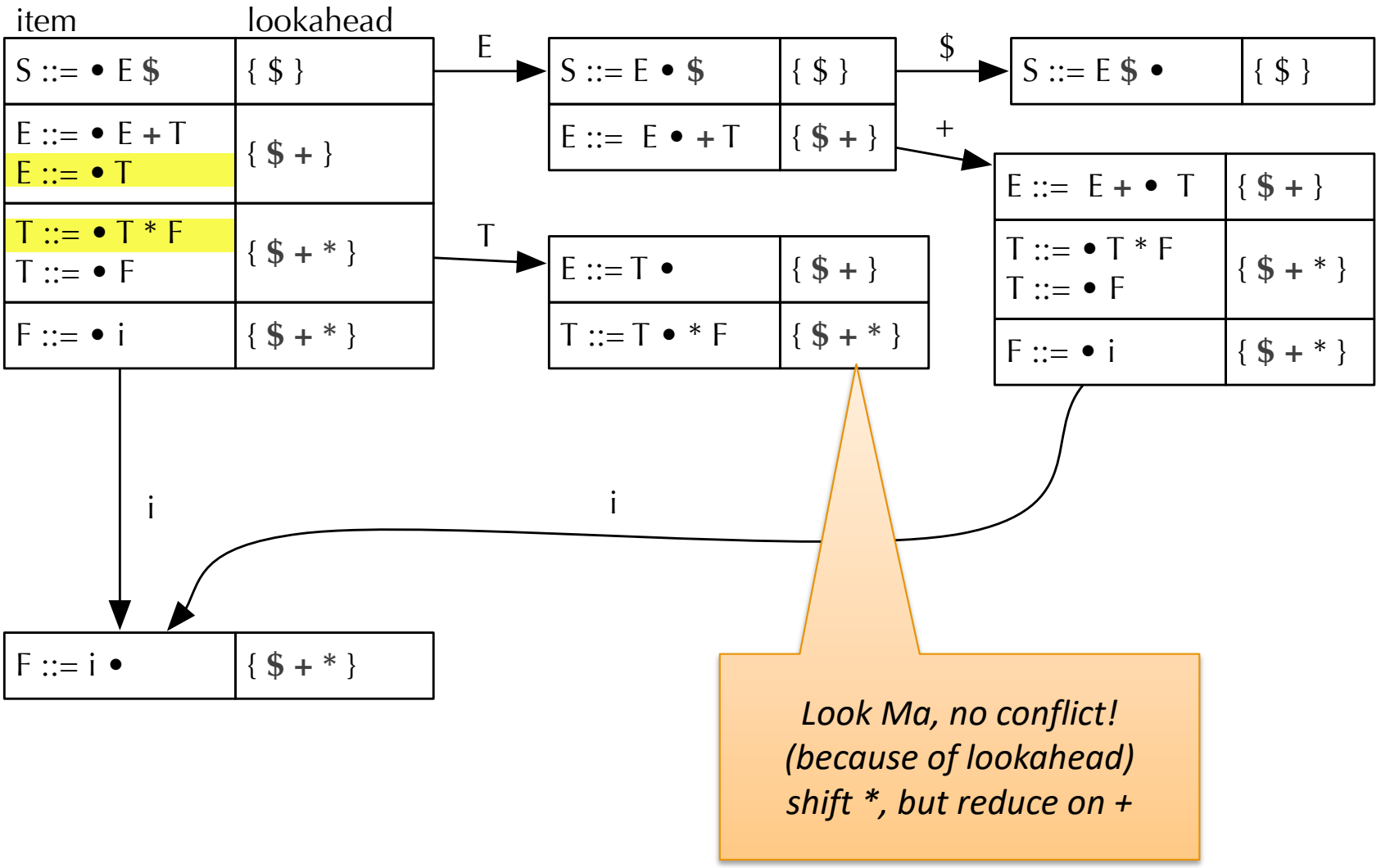
Shift i

CFSM



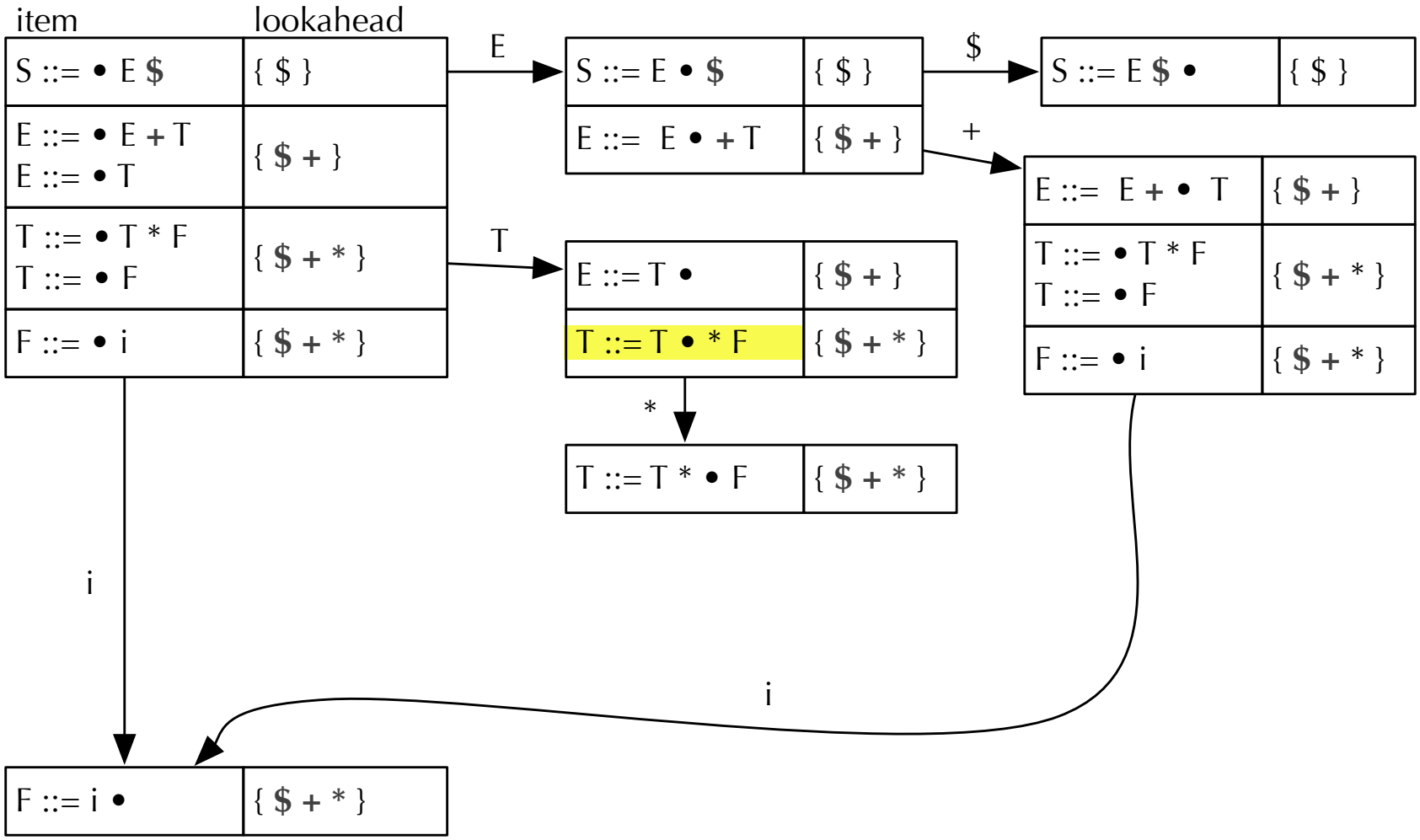
Goto T

CFSM


item
lookahead
S ::= • E $
{ $ }
E ::= • E + T
E ::= • T
{ $ + }
T ::= • T * F
T ::= • F
{ $ + * }
F ::= • i
{ $ + * }
E
S ::= E • $
{ $ }
E ::= E • + T
{ $ + }
$
S ::= E $ •
{ $ }
+
E ::= E + • T
{ $ + }
T ::= • T * F
T ::= • F
{ $ + * }
F ::= • i
{ $ + * }
T
E ::= T •
{ $ + }
T ::= T • * F
{ $ + * }
i
i
F ::= i •
{ $ + * }
Look Ma, no conflict!
(because of lookahead)
shift *, but reduce on +

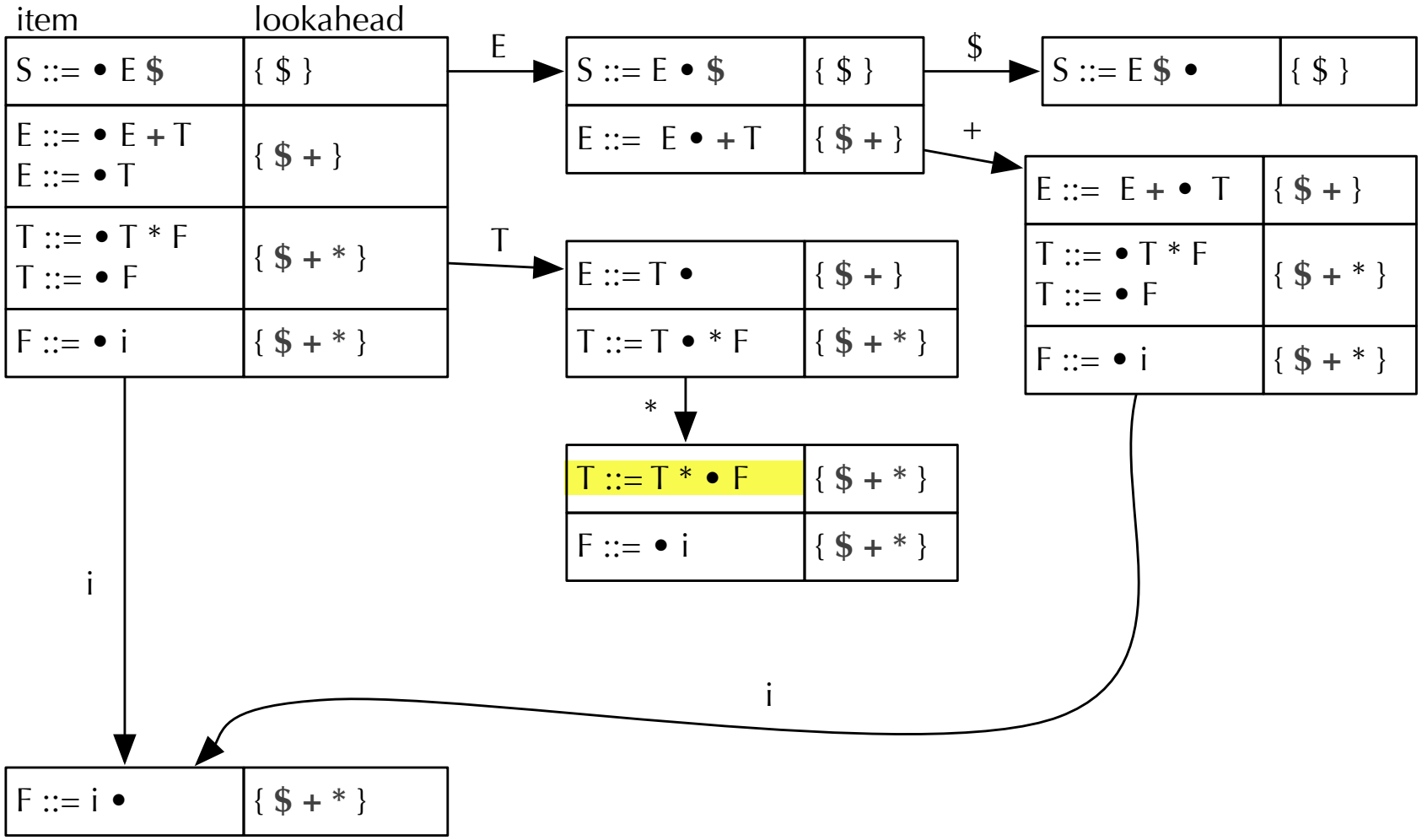
Shift *

CFSM


item
lookahead
S ::= • E $
{ $ }
E ::= • E + T
E ::= • T
{ $ + }
T ::= • T * F
T ::= • F
{ $ + * }
F ::= • i
{ $ + * }
E
S ::= E • $
{ $ }
E ::= E • + T
{ $ + }
$
S ::= E $ •
{ $ }
+
E ::= E + • T
{ $ + }
T ::= • T * F
T ::= • F
{ $ + * }
F ::= • i
{ $ + * }
T
E ::= T •
{ $ + }
T ::= T • * F
{ $ + * }
*
T ::= T * • F
{ $ + * }
i
i
F ::= i •
{ $ + * }

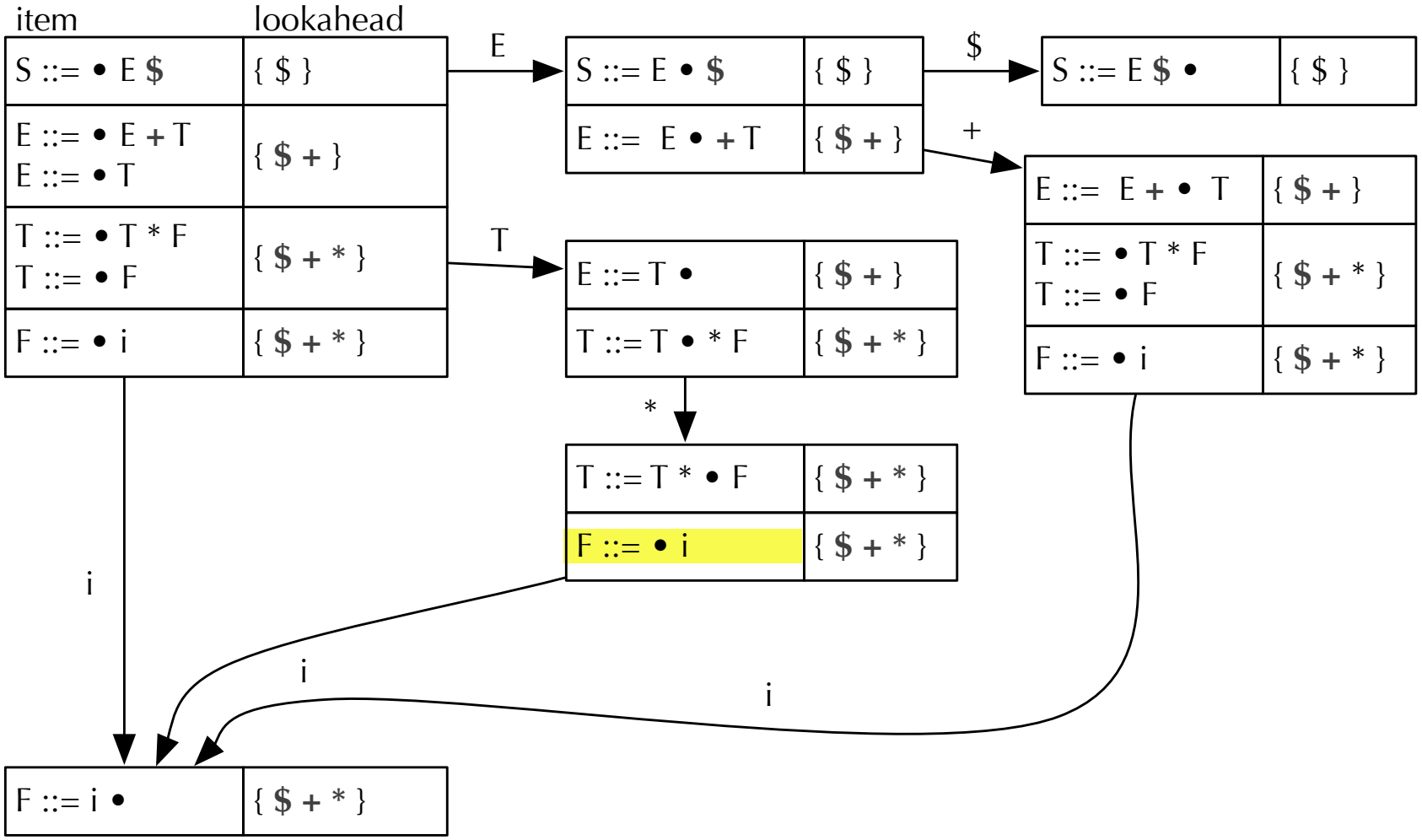
Expand T

CFSM

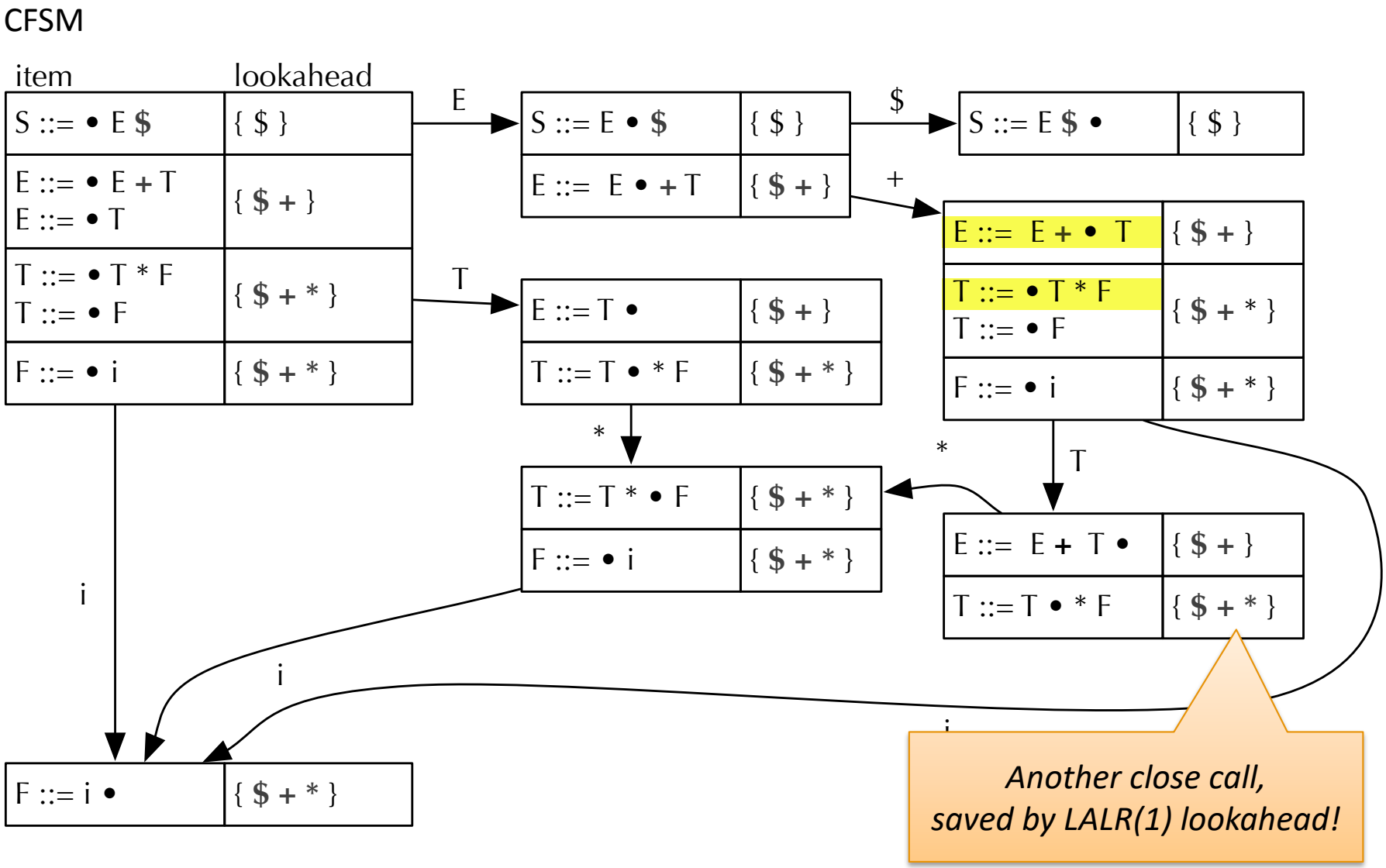


Shift i

CFSM

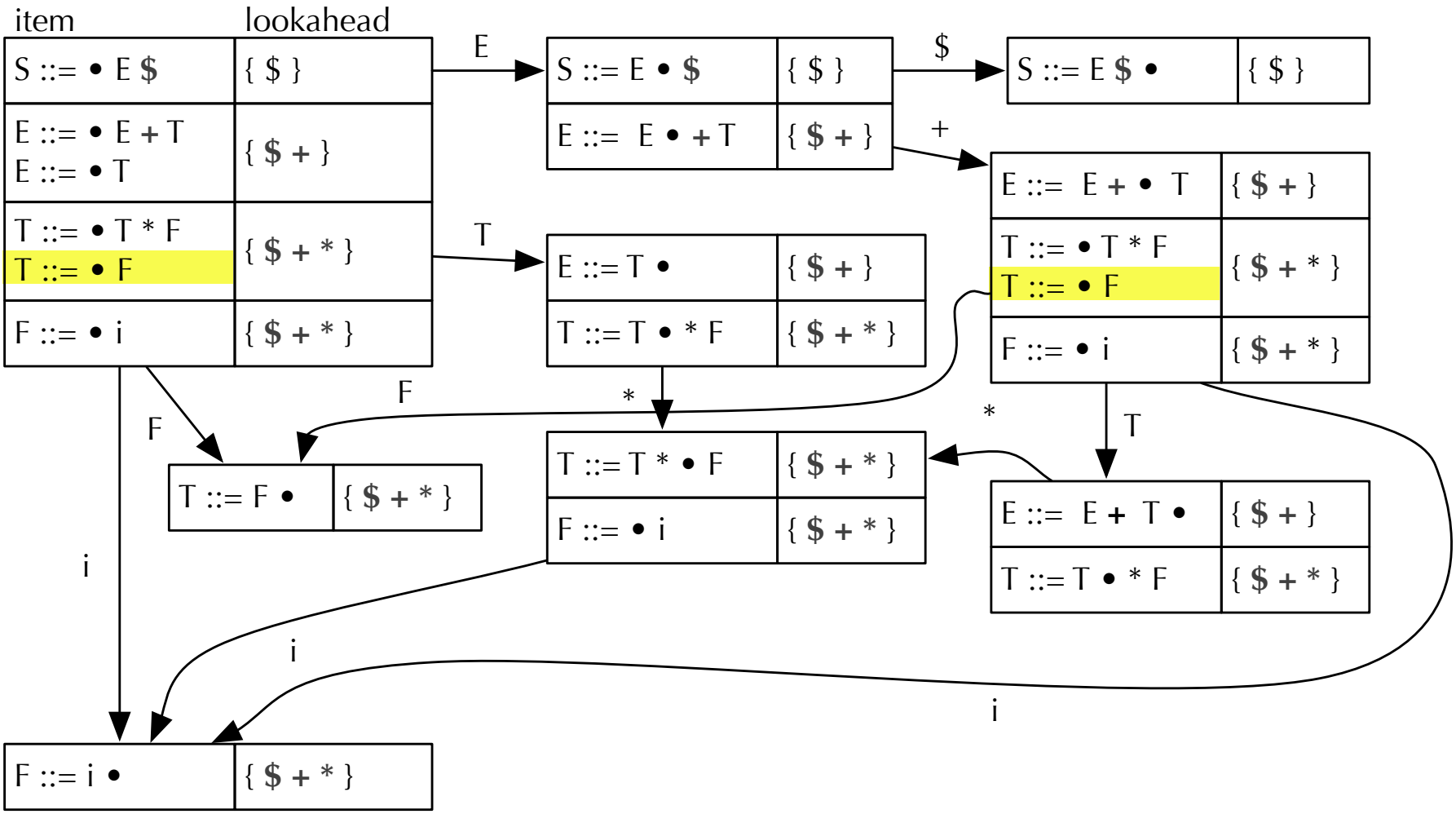

item
lookahead
S ::= • E $
{ $ }
E ::= • E + T
E ::= • T
{ $ + }
T ::= • T * F
T ::= • F
{ $ + * }
F ::= • i
{ $ + * }
E
S ::= E • $
{ $ }
E ::= E • + T
{ $ + }
$
S ::= E $ •
{ $ }
+
E ::= E + • T
{ $ + }
T ::= • T * F
T ::= • F
{ $ + * }
F ::= • i
{ $ + * }
T
E ::= T •
{ $ + }
T ::= T • * F
{ $ + * }
*
T ::= T * • F
{ $ + * }
F ::= • i
{ $ + * }
i
i
i
F ::= i •
{ $ + * }

Goto T


CFSM
item
lookahead
S ::= • E $
{ $ }
E ::= • E + T
E ::= • T
{ $ + }
T ::= • T * F
T ::= • F
{ $ + * }
F ::= • i
{ $ + * }
E
S ::= E • $
{ $ }
E ::= E • + T
{ $ + }
$
S ::= E $ •
{ $ }
+
E ::= E + • T
{ $ + }
T ::= • T * F
T ::= • F
{ $ + * }
F ::= • i
{ $ + * }
T
E ::= T •
{ $ + }
T ::= T • * F
{ $ + * }
*
T ::= T * • F
{ $ + * }
F ::= • i
{ $ + * }
*
T
E ::= E + T •
{ $ + }
T ::= T • * F
{ $ + * }
i
i
i
F ::= i •
{ $ + * }
Another close call,
saved by LALR(1) lookahead!

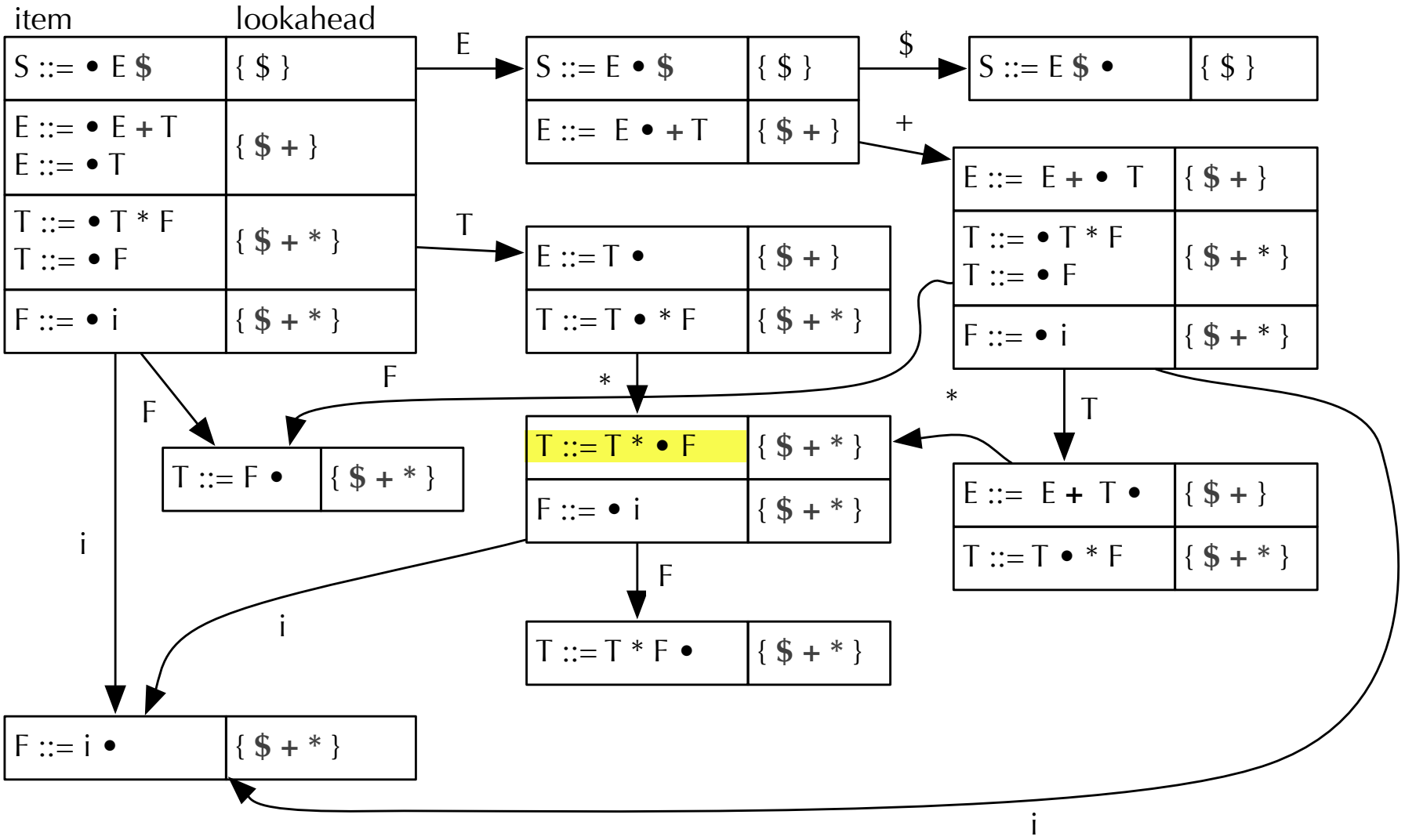
Goto F

CFSM


item
lookahead
S ::= • E $
{ $ }
E ::= • E + T
E ::= • T
{ $ + }
T ::= • T * F
T ::= • F
{ $ + * }
F ::= • i
{ $ + * }
E
S ::= E • $
{ $ }
E ::= E • + T
{ $ + }
$
S ::= E $ •
{ $ }
+
E ::= E + • T
{ $ + }
T ::= • T * F
T ::= • F
{ $ + * }
F ::= • i
{ $ + * }
T
E ::= T •
{ $ + }
T ::= T • * F
{ $ + * }
F
F
T ::= F •
{ $ + * }
*
*
T
T ::= T * • F
{ $ + * }
F ::= • i
{ $ + * }
E ::= E + T •
{ $ + }
T ::= T • * F
{ $ + * }
i
i
i
F ::= i •
{ $ + * }

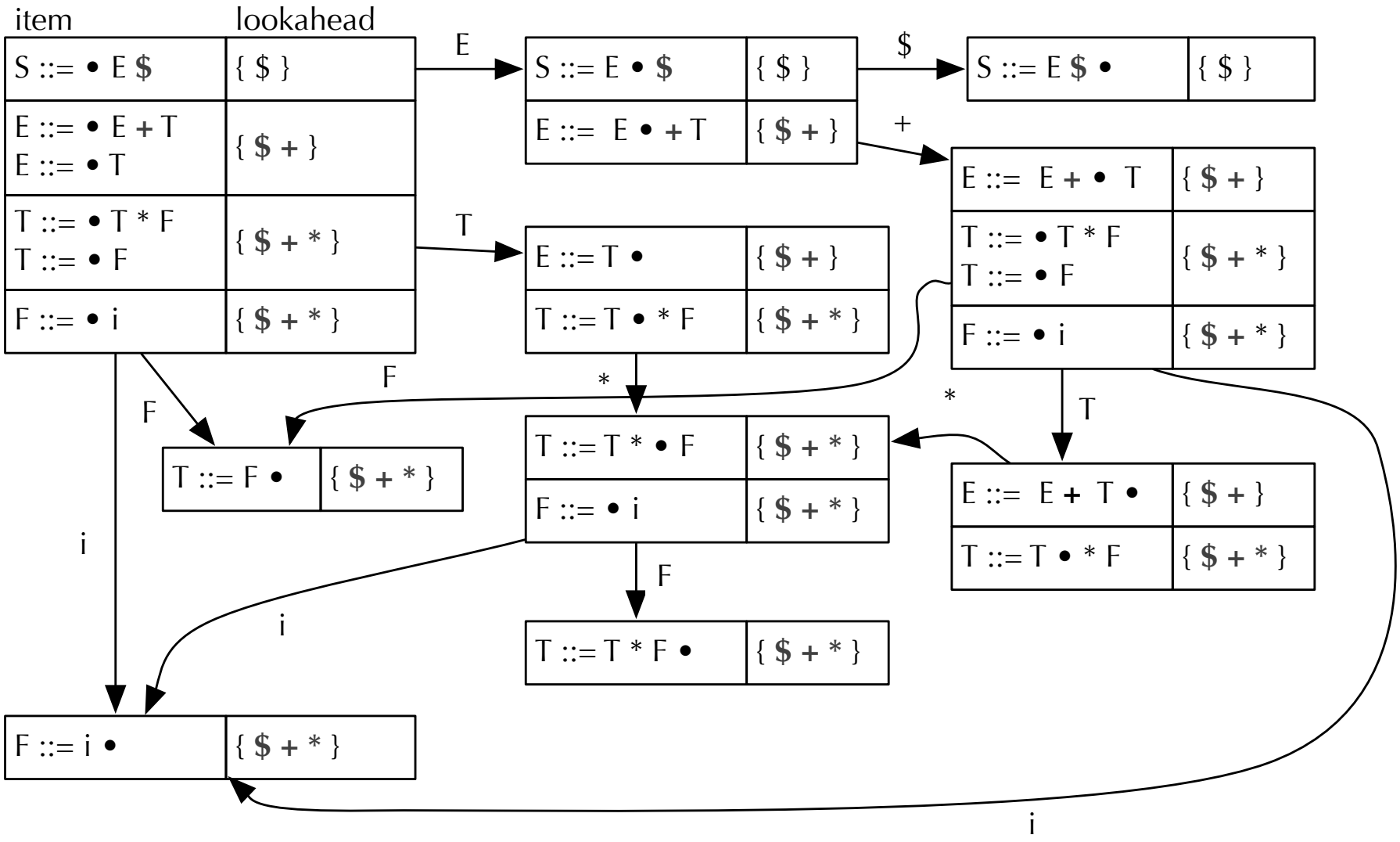
Goto F

CFSM


item
lookahead
S ::= • E $
{ $ }
E ::= • E + T
E ::= • T
{ $ + }
T ::= • T * F
T ::= • F
{ $ + * }
F ::= • i
{ $ + * }
E
S ::= E • $
{ $ }
E ::= E • + T
{ $ + }
$
S ::= E $ •
{ $ }
+
E ::= E + • T
{ $ + }
T ::= • T * F
T ::= • F
{ $ + * }
F ::= • i
{ $ + * }
T
E ::= T •
{ $ + }
T ::= T • * F
{ $ + * }
F
F
T ::= F •
{ $ + * }
*
T ::= T * • F
{ $ + * }
F ::= • i
{ $ + * }
*
T
E ::= E + T •
{ $ + }
T ::= T • * F
{ $ + * }
F
T ::= T * F •
{ $ + * }
i
i
F ::= i •
{ $ + * }
i

Complete? No conflicts? Then the grammar is LALR(1)

CFSM



Notes …

To show it is LALR(1), we need the whole CFSM (or at least the non-boring parts)

To show it is *not* LALR(1), we just need a path to one state with shift/reduce or reduce/reduce conflict

Shift/reduce conflicts indicate we haven't resolved priority

Reduce/reduce suggest deeper problems of ambiguity. There is no default resolution