2202-COL226 Major

Utkarsh Singh

TOTAL POINTS

30.5 / 80

QUESTION 1

1 **0 / 8**

+ 1.5 pts Part a Case 1

If

$$\mathscr{E} [e] \; (\mathscr{C}[c] \; s) = \top$$

Then

$$\mathscr{C}[\textbf{repeat } c \textbf{ until } e] \; s = \mathscr{C}[c] \; s$$

+ 1.5 pts Part a Case 2

If

$$\mathscr{E} [e] \; (\mathscr{C}[c] \; s) = \bot$$

Then

$$\mathscr{C}[\textbf{repeat } c \textbf{ until } e] \; s = \mathscr{C}[\textbf{repeat } c \textbf{ until } e] \; (\mathscr{C}[c] \; s)$$

+ 2.5 pts Part b Rule 1

If

$$\gamma \vdash <\sigma, c > \rightarrow^1_c \sigma' \qquad \gamma \vdash <\sigma', e > \rightarrow_e \top$$

Then

$$\gamma \vdash <\sigma, \textbf{repeat } c \textbf{ until } e> \rightarrow^1_c \sigma'$$

+ 2.5 pts Part b Rule 2

If

$$\gamma \vdash <\sigma, c > \rightarrow^1_c \sigma' \quad \gamma \vdash <\sigma', e > \rightarrow_e \bot$$

$$\gamma \vdash <\sigma', \textbf{repeat } c \textbf{ until } e> \rightarrow^1_c \sigma''$$

Then

$$\gamma \vdash <\sigma, \textbf{repeat } c \textbf{ until } e> \rightarrow^1_c \sigma''$$

✓ **+ 0 pts** *None*

QUESTION 2

2 **5 / 21**

+ 1 pts Rule 1

+ 3 pts Rule 2

+ 3 pts Rule 3

+ 3 pts Rule 4

+ 5 pts Rule 5

+ 5 pts Rule 6

+ 1 pts Rule 7

- 5 pts Did not properly emit code (IR) / Provided an execution instead of IR

✓ **+ 0 pts** *Incorrect / Not Attempted*

+ 5 *Point adjustment*

Some basic understanding.

QUESTION 3

3 **6 / 11**

✓ **+ 4 pts** *Part a Correct*

- 1 pts Part a minor mistake

+ 3.5 pts Part b (-+n = -n) Correct

+ 3.5 pts Part b (--n = +n) Correct

+ 3 pts Part b partially correct

+ 0 pts Incorrect / Not Attempted

+ 2 *Point adjustment*

QUESTION 4

4 **0 / 8**

- 0 pts All steps correct, justification present, correct principal type

- 1 pts Missing justification (Inference rule used in each step i.e var/app/abstraction/pair)

- 0.5 pts Minor error in applying abstraction type inference rule: Not modifying context $${\frac{\Gamma,x\mathbin{:}\sigma\vdash e\mathbin{:}\tau}{\Gamma\vdash (\lambda x\mathbin{:}\sigma.~e)\mathbin{:}(\sigma \to \tau)}}$$

- 2 pts Minor errors (Error in notation, missing justifications,minor mistakes in applying inference rules)

- 4 pts Not formal, no proper notation, written verbally

✓ **- 8 pts** *Fully Incorret/Unattempted*

QUESTION 5

5 **10 / 12**

✓ **+ 4 pts** $**N\ 1 \mid\mid_\beta \leftarrow L \rightarrow 1 \mid\mid_\beta M \Rightarrow \exists P : N \rightarrow 1 \mid\mid_\beta P\ 1 \mid\mid_\beta \leftarrow M**$

case $L \equiv x \in V$.

case $L \equiv alpha\ M$

case $L \equiv alpha\ N$

case $M \equiv alpha\ N$

✓ **+ 2 pts** $**N \not\equiv_\alpha L \not\equiv_\alpha M \not\equiv_\alpha N**$

Case $L \equiv \lambda x[L1\ ].$

✓ **+ 2 pts** $**N \not\equiv_\alpha L \not\equiv_\alpha M \not\equiv_\alpha N**$

Case $L \equiv (L1\ L2).$

+ 4 pts $**N \not\equiv_\alpha L \not\equiv_\alpha M \not\equiv_\alpha N**$

Main case $L \equiv (\lambda x[L1\ ]\ L2)$

it has 4 sub cases

+ 0 pts Not attempted or Wrong

+ 2 *Point adjustment*

For stating property abstraction2 and subcases are not complete

QUESTION 6

6 **9.5 / 20**

Part a

✓ **+ 1 pts** *Correct Example*

✓ **+ 1.5 pts** *Correct 1st Parse Tree*

✓ **+ 1.5 pts** *Correct 2nd Parse Tree*

+ 0 pts Incorrect/Unattempted

Part b

+ 10 pts Correctly showed all **three** shift/reduce conflicts

+ 7 pts Correctly showed atleast **two** shift/reduce conflicts

✓ **+ 4 pts** *Correctly showed **one** shift/reduce conflicts*

- 3 pts Showed a reduce/reduce conflicts.

+ 0 pts Incorrect/Not Attempted.

Part c

✓ **+ 1.5 pts** *Correct Associativity*

+ 1.5 pts Correct Precedence

+ 3 pts Removal of left-recursion

+ 0 pts Incorrect/Unattempted

Part C:

Step 1 (Converting into unambiguous with Correct Associativity and Precedence):

S->T$

T->T^U|T_U|U

U->UV|V

V->{T}|a

Step 2 (Removal of left recursion):

S->T$

T->UT'

T'->^UT'|_UT'|eps

U->VU'

U'->VU'|eps

V->{T}|a

Name: Utkarsh Singh LoginId: CS121058@cse.iitd.ac.in 1

COL226: Programming Languages

II semester 2022-23

Sun 07 May 2023 | Major | LH325 | 120 minutes | Max Marks 80

Instructions.
1. Answer only in the space provided for each question in the question paper itself.
2. Write your name and your **IITD Loginid** on the top line of every page
3. No extra sheets will be provided. You may do your rough work in the separately provided sheets.
4. Answers will be judged for correctness, efficiency and elegance.
5. If there are minor mistakes in the question, correct them explicitly and answer the question accordingly.
6. If the question is totally wrong, give adequate reasons why it is wrong with detailed counter-examples.

---

1. **[3 + 5 = 8 marks]** Assume the WHILE programming language is extended to include a command in which the command c is repeatedly executed an terminates only if the boolean expression e evaluates to *true*.

   (a) Define $\mathscr{C}$[**repeat** c **until** e] $\stackrel{df}{=} \lambda s[\ldots]$

   (b) Define operational rules for **repeat** c **until** e.

2. **[4+10+6 = 21 marks]** Consider the following augmented grammar for a language which has both an **if-then** and an **if-then-else** construct. The grammar resolves the dangling-else ambiguity in favour of the **else**-clause being associated with the textually closest occurring condition that has not already been matched by an earlier **else**-clause. The tokens **if**, **then** and **else** have been abbreviated by the symbols `i`, `t` and `e` respectively. b denotes a boolean expression and a denotes any other expression.

$G = \langle \{S, I, U, M\}, \{\mathtt{i}, \mathtt{t}, \mathtt{e}, b, a\}, P, S \rangle$ where P the set of productions is given by

1. $S \rightarrow I\,\$$
2. $I \rightarrow U$
3. $I \rightarrow M$
4. $U \rightarrow \mathtt{i}\ b\ \mathtt{t}\ I$
5. $U \rightarrow \mathtt{i}\ b\ \mathtt{t}\ M\ \mathtt{e}\ U$
6. $M \rightarrow \mathtt{i}\ b\ \mathtt{t}\ M\ \mathtt{e}\ M$
7. $M \rightarrow a$

Define syntax-directed translation rules for this grammar.

AST grammar:

$S \rightarrow I\,\$$ $\quad (S^e(I^i))$

$I \rightarrow U$ $\quad (I_1(U))$

$I \rightarrow M$ $\quad (I_2(M))$

$U \rightarrow \mathtt{i}\ b\ \mathtt{t}\ I$ $\quad (U_1(I))$

$U \rightarrow \mathtt{i}\ b\ \mathtt{t}\ M\ \mathtt{e}\ U$ $\quad (U_2(M, U))$

$M \rightarrow \mathtt{i}\ b\ \mathtt{t}\ M\ \mathtt{e}\ M$ $\quad (M_1(M', M''))$

$M \rightarrow a$ $\quad (M_2(a))$

Datatype:

datatype of S = datatype of I

I_1 = datatype of U

I_2 = datatype of M

U_1 = datatype of I

U_2 = datatype of M * datatype of U

M_1 = datatype of M * datatype of M

M_2 = datatype of a

Evaluator:

~~M_2 = return a~~

~~M_2 (a) return a~~

$S(I) = I_1(I)$;

$I_1(V) =$

3. [4 + 7 = 11 marks] Given

$\underline{0}$	$\stackrel{df}{=}$	$\lambda f\ x[x]$	$\underline{n+1}$	$\stackrel{df}{=}$	$\lambda f\ x[(f\ (f^n\ x))]$
Pair	$\stackrel{df}{=}$	$\lambda x\ y\ p[(p\ x\ y)]$			
Fst	$\stackrel{df}{=}$	$\lambda p[(p\ \text{True})]$	Snd	$\stackrel{df}{=}$	$\lambda p[(p\ \text{False})]$
Convert	$\stackrel{df}{=}$	$\lambda x[(\text{Pair}\ x\ \underline{0})]$	Neg	$\stackrel{df}{=}$	$\lambda x[(\text{Pair}\ (\text{Snd}\ x)\ (\text{Fst}\ x))]$

Church defined signed integers $\underline{+n}$ and $\underline{-n}$ for each Church numeral $\underline{n}$, by first Converting a Church numeral $\underline{n}$ into $\underline{+n}$ and then Negating $\underline{+n}$ to obtain $\underline{-n}$.

Prove that in this representation

(a) $+0$ and -0 have the same representation, i.e $\underline{+0} =_\beta \underline{-0}$

(b) for each signed integer $\pm n$, $- \pm n =_\beta \mp n$

a) $\underline{+0} = (\lambda n\ [(\text{Pair}\ n\ \underline{0})]\ \lambda f\ x[x])$

$= (\text{Pair}\ \underline{0}\ \underline{0}) = \lambda p\ [(p\ \underline{0}\ \underline{0})]$

$\underline{-0} = \lambda n\ [(\text{Pair}\ (\text{Snd}\ n)\ (\text{fst}\ n))]$ +0

$= (\text{Pair}\ (\text{Snd}\ +0)\ (\text{fst}\ +0))$

$= (\text{Pair}\ (+0\ \text{False})\ (+0\ \text{True}))$

$= (\lambda x y p[(p\ x\ y)]\ (+0\ \text{false})\ (+0\ \text{True}))$

$= \lambda p\ [(p\ (+0\ \text{false})\ (+0\ \text{True}))]$

$= \lambda p\ [(p\ (\text{False}\ \underline{0}\ \underline{0})\ (\text{True}\ \underline{0}\ \underline{0}))]$

$= \lambda p\ [(p\ \underline{0}\ \underline{0})] = \underline{+0}$

hence proved

b) Case I: $\underline{+n} =_\beta \underline{-n}$

Convert Neg $\underline{n} =_\beta$ Neg Convert $\underline{n}$

Convert Neg $\underline{n}$ $=_\beta$ Neg Convert $\underline{n}$

Convert Neg $\underline{n}$ = $\lambda x\,[(\text{Pair } x\ \underline{0})]$ [(Pair (Snd $\underline{n}$) (fst $\underline{n}$))]

= $\lambda x\,[(\text{Pair } x\ \underline{0})]$ [(Pair ($\underline{n}$ false) ($\underline{n}$ true))]

= $\lambda x\,[\lambda p[(p\ x\ \underline{0})]]$ [$\lambda p\,(p\ (\underline{n}$ false) ($\underline{n}$ true))]

= $\lambda p\,[(p\ \lambda p\,[(p\ (\underline{n}$ false) ($\underline{n}$ true))] $\underline{0}$)]

= $\lambda p\,[(p\ 10$

Neg Convert $\underline{n}$ = ~~$\lambda x\,[(\text{Pair (Snd } x)\ (\text{fst } x))]$~~

Neg (Pair $\underline{n}$ $\underline{0}$)

= $\lambda x\,[(\text{Pair (Snd } x)\ (\text{fst } x)]$ (Pair $\underline{n}$ $\underline{0}$)

= $\lambda x\,[(\text{Pair (Snd } x)\ (\text{fst } x)]$ [$\lambda p\,[p\ \underline{n}\ \underline{0}$]

= (Pair [Snd

4. [8 marks]

We have seen that the function curry2 $\stackrel{df}{=} \lambda x[\lambda y[\lambda z[(x\ \langle y, z\rangle)]]]$ allows one to curry a binary operator. Give a formal proof of the principal type of the combinator curry2 using the inference rules. Assume that the rule for pairing is, if $x : \sigma$ and $y : \tau$ then $\langle x, y\rangle : \sigma * \tau$.

Justify each step to get full marks.

Soln ~~$(\lambda x[\lambda y[\lambda z[(x\ y\ z)]]]\ L)$~~

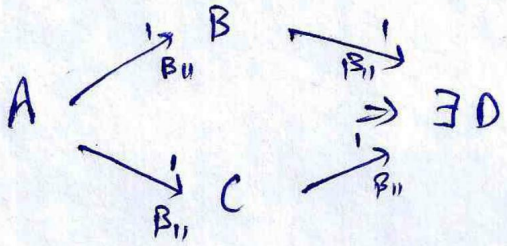
$= \lambda x[\lambda y[\lambda z[(x \langle y, z\rangle)]]]\ L$

$= \lambda y[\lambda z[(L \langle y, z\rangle)]]$

5. **12 marks** Prove the following theorem.

Theorem 0.1 $\xrightarrow{1}_{||\beta}$ *has the diamond property.*

Sol") We will prove the diamond property by applying induction on the structure of terms of A.



Case I) $A \equiv_\lambda B$, ~~$A \not\equiv_\lambda C$~~, $A \equiv_\lambda C$, this can be shown trivially by taking $D = A$

Case II) $A \equiv_\lambda B$, $A \not\equiv_\lambda C$,

$A \xrightarrow[\beta_{||}]{1} C \Rightarrow B \xrightarrow[\beta_{||}]{1} C$, and $C \xrightarrow[\beta_{||}]{} C$

$\therefore D = C$.

Case III) $\lambda x[A_1] = A$, $\lambda x[A_1] \xrightarrow[\beta_{||}]{1} \lambda x[B_1]$

$\downarrow \beta_{||}$

$\lambda x[C_1]$

$\because$ we assumed ~~above~~, [induction hypothesis] $A_1 \xrightarrow{1} B_1$, $B_1 \xrightarrow{\beta_{||}} \exists D_1$, $A_1 \xrightarrow{1} C_1$, $C_1 \xrightarrow{\beta_{||}} D_1$

$\therefore$ if $D = \lambda x[D_1]$, hence proved.

Case IV) $(A_1\ A_2)$, this will comprise of 3 subcases, all three of the subcases will be exhaustive in nature.

case1) $(\lambda x[A_1'], y)$ i.e A_2 is just a free variable.

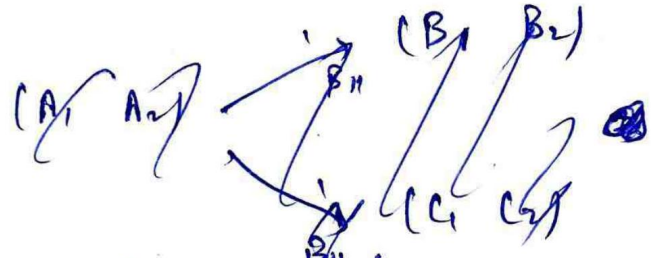
$(\lambda x[A_1'], y) \xrightarrow[\beta_{||}]{1} A_{||}'$

i.e after applying a $\beta_{||}$ reduction, it gets reduced to one of the cases which we proved above.

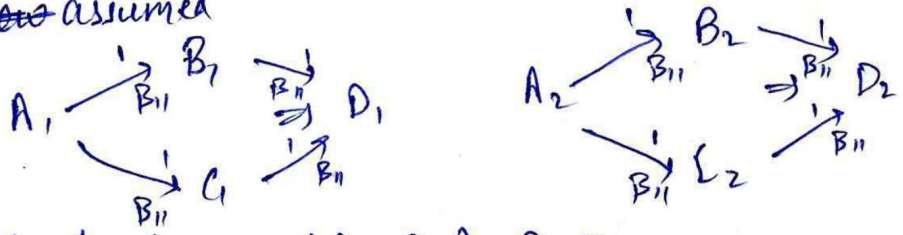
~~$A_1 \xrightarrow{\beta_{||}}$~~ $[\lambda x[A_1'], y] \xrightarrow[\beta_{||}]{1} A_{||}'$, $A_{||}' \xrightarrow{\beta_{||}} B_1$, $B_1 \xrightarrow{\beta_{||}} \exists D$, $A_{||}' \xrightarrow{\beta_{||}} C_1$, $C_1 \xrightarrow{\beta_{||}} D$

hence proved

case 2) $(A_1\ A_2)$, where A_1 and A_2 are not abstractions.



we ~~knew~~ assumed

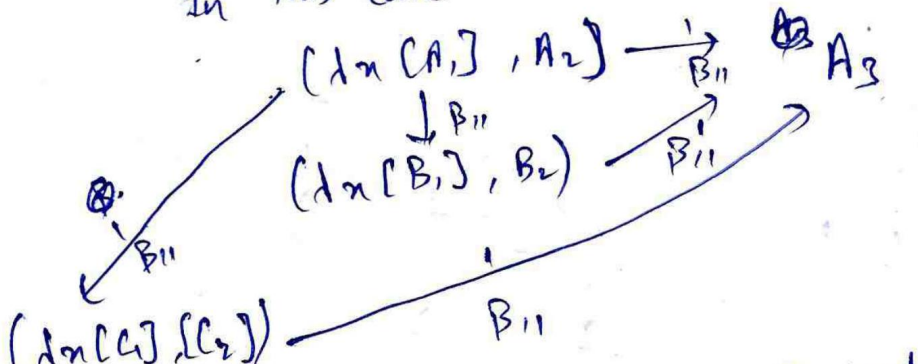


∴ if we choose $(D_1, D_2) = D$ then

$(C_1\ C_2) \xrightarrow{\beta_{11}} (D_1, D_2)$ and $(B_1, B_2) \xrightarrow{\beta_{11}} (D_1, D_2)$

case 3) $(\lambda x\,[A_1], A_2)$ where A_2 is a general λ term.

In this case



~~In any of the thr.~~ here B and C can by any of the terms above.

∴ if we choose $D = A_3$ then $B \xrightarrow{\beta_{11}} D$

$C \xrightarrow{\beta_{11}} D$

hence proved.

6. [4+10+6 = 20 marks]

Consider the following ambiguous augmented grammar for generating text with subscripts and superscripts for formulas in a mathematical text-processor, like LaTeX. Assume

- **a** is a piece of text and that sequences of **a** are used for the main text as well as those of superscripts and subscripts.
- "^" is the infix superscript operator, "_" the infix subscript operator, "{" and "}" is the pair of parenthesis that users may use to override precedence and associativity to make their intent unambiguous.
- Both "^" and "_" are left-associative and have equal precedence as operators.
- juxtaposition has higher precedence than both "^" and "_" and is also left-associative.
- $ is the end-of-text marker.

$$S \rightarrow T\$$$
$$T \rightarrow T\hat{}T \mid T\_T \mid \{T\} \mid TT \mid \mathbf{a}$$

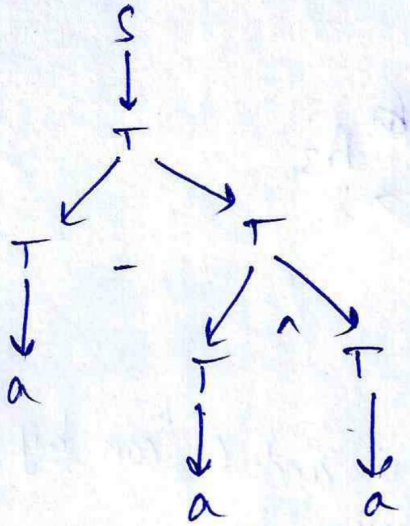
(a) Give an example of a string that has at least two distinct parse trees and construct the parse trees.

(b) Design an LR(0) parsing table for the above grammar and determine all the conflicts.

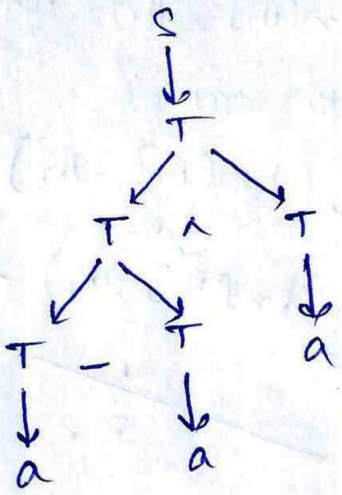
(c) Redesign the grammar so that it is unambiguous, satisfies the above associativity and precedence rules and is LL(1) parseable.

Sol^n a) Let us consider a string a_a^a

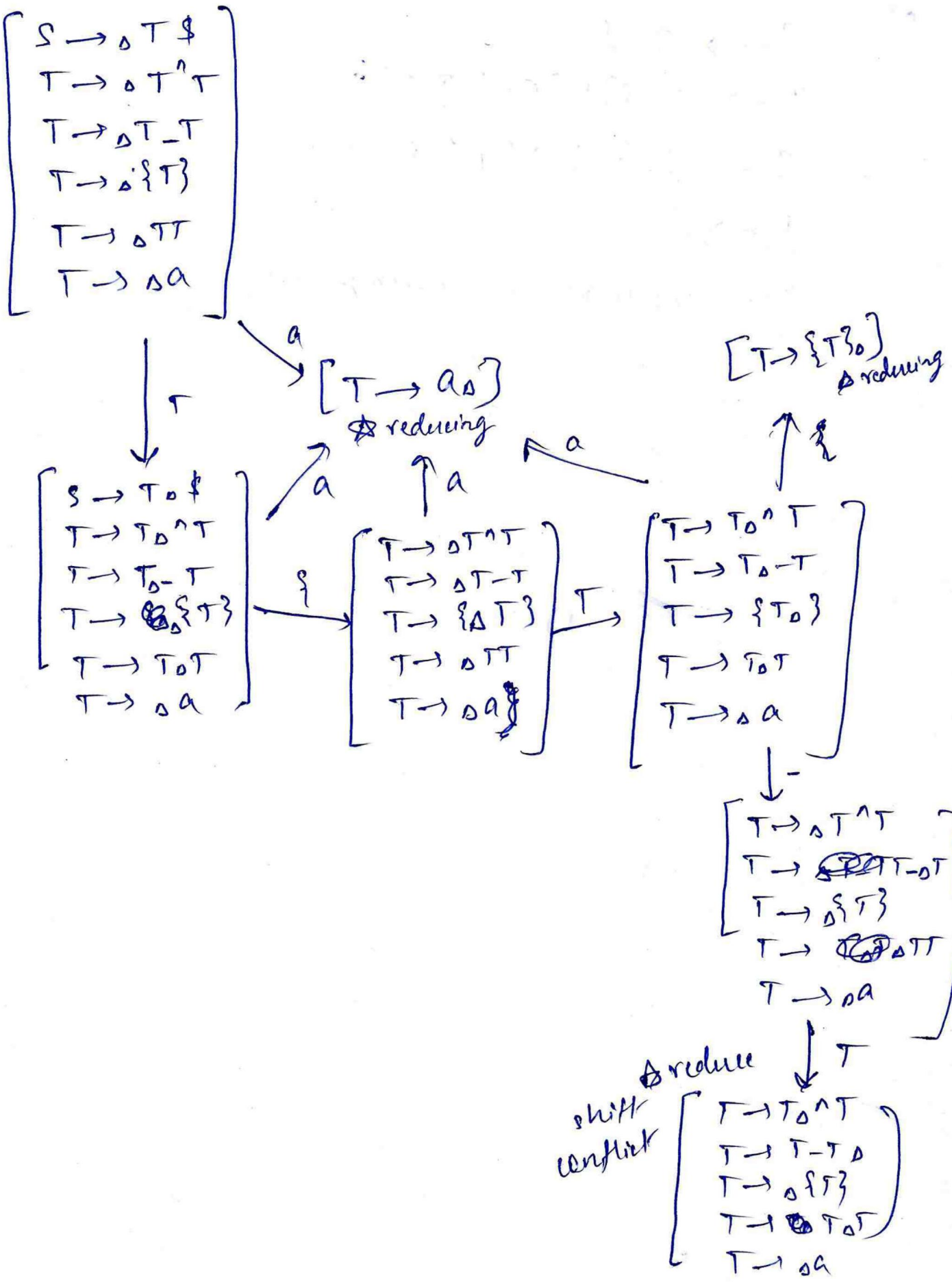
Parse tree 1



Parse tree 2



∴ The gramanar has two different parse trees
∴ it is ambiguous.


S → ∘T $
T → ∘T^T
T → ∘T_T
T → ∘{T}
T → ∘TT
T → ∘a
a
T
[T → a∘]
reducing
a
a
[T → {T}∘]
reducing
S → T∘$
T → T∘^T
T → T∘_T
T → {∘T}
T → T∘T
T → ∘a
{
T → ∘T^T
T → ∘T_T
T → {∘T}
T → ∘TT
T → ∘a
T
T → T∘^T
T → T∘_T
T → {T∘}
T → T∘T
T → ∘a
T → ∘T^T
T → T_T∘
T → ∘{T}
T → ∘TT
T → ∘a
reduce
shift conflict
T
T → T∘^T
T → T_T∘
T → ∘{T}
T → {T∘}T
T → ∘a

c)

$S \longrightarrow T\ \$$

$T \longrightarrow T\{S\} \mid \{S\} \mid TS \mid S$

$S \longrightarrow S\_R \mid S\hat{}R \mid R$

$R \longrightarrow a$

The grammar is unambiguous.