Homework 3

Kai Qu
kq4ff

November 2, 2019

1 Let $s \rightarrow b$ denote the right arc where s is the head and b is the dependent, and $s \leftarrow b$ denote the left arc where s is the dependent and b is the head.

Algorithm:

> Given stack S, input buffer (queue) B, list of actions $A = empty\ list$, Dependency (Dictionary $(Key : int, Value : list)$ D
> $flag = False$
> # the case when the first two actions cannot be SHIFT
> if $len(B) < 2$:
>   return actions
> else:
> $S.push(B.dequeue)$
> $S.push(B.dequeue)$
> $A.append('SHIFT')$
> $A.append('SHIFT')$
> while $flag\ is\ False$:
> $flag = True$
>   if $len(S) > 1$:
> $b =$ the first element on the stack
> $s =$ the second element on the stack
>   # update dependencies
>   for k in $D.keys$:
> $D[k].remove(b)$
> $D[s].remove(s)$
>   # left arc action
>   if $s \leftarrow b$ and $D[s]$ is empty:
> $S.pop$
> $S.pop$
> $S.push(b)$
> $A.append('LEFTARC')$
> $flag = False$

```
# right arc action
    else if s → b and D[b] is empty:
        S.pop
        S.pop
        S.push(s)
        A.append('RIGHTARC')
        flag = False
# shift action
if flag is True and len(B) > 0
    b = B.dequeue
    S.push(b)
    A.append('SHIFT')
    flag = False
```

if $len(S)$ is not 1:

  return "non-projective case"

else:

  return actions

2 (a) Actions:

SHIFT:
1. Dequeue one element from the input buffer
2. Push the element on the top of the stack

UNARY-REDUCE:
1. Pop 1 element, determined by the production rule, from the top of the stack
2. Push the element, produced by the production rule, on the top of the stack

BINARY-REDUCE:
1. Pop 2 elements, determined by the production rule, from the top of the stack
2. Push the element, produced by the production rule, on the top of the stack

(b) Procedure:

1. SHIFT the first token *the* onto the stack.
2. UNARY-REDUCE the top element *the* into DT by the production rule $DT \to the$.
3. SHIFT the second token *man* onto the stack.
4. UNARY-REDUCE the first element on the stack *man* into NN by the production rule $NN \to man$
5. BINARY-REDUCE the top two elements on the stack DT NN into NP by the production rule $NP \to DT\ NN$.
6. SHIFT the third token *sleeps* onto the stack.

7. UNARY-REDUCE the top element on the stack *sleeps* into Vi by the production rule $Vi \rightarrow sleeps$.
8. UNARY-REDUCE the top element on the stack Vi into VP by the production rule $VP \rightarrow Vi$.
9. BINARY-REDUCE the top two elements on the stack NP and VP into S by the production rule $S \rightarrow NP\ VP$. The stack now only contains S. Since the input buffer is empty, this is an accepting state.