

Corrigé DS4 - type Mines

Déduction naturelle

① on remarque alors que :

$$\mu(\perp) = \emptyset$$

$$\mu(A \wedge B) = \mu(A) \cap \mu(B) = \emptyset \text{ ssi } \mu(A) = \emptyset \text{ ou } \mu(B) = \emptyset$$

$$\mu(A \vee B) = \mu(A) \cup \mu(B) = \emptyset \text{ ssi } \mu(A) = \emptyset \text{ et } \mu(B) = \emptyset$$

$$\mu(A \rightarrow B) = \overline{\mu(A)^c \cup \mu(B)} = \emptyset \text{ ssi } \mu(B) = \emptyset \text{ et } \mu(A) = IR$$

Ainsi en posant $\mu(A) = \emptyset$ ssi $\mathcal{V}(A) = \text{faux}$
et $\mu(A) = IR$ ssi $\mathcal{V}(A) = \text{vrai}$

on retrouve exactement la sémantique booléenne standard.

②

$$\frac{\frac{\frac{\frac{}{\Gamma \vdash (A \vee B) \wedge \neg A} \text{ } \wedge}{\Gamma \vdash A \vee B} \text{ } \vee \wedge}{\Gamma \vdash (A \vee B) \wedge \neg A} \text{ } \wedge \quad \frac{\frac{\frac{}{\Gamma, A \vdash A; \Gamma, A \vdash \neg A} \text{ } \wedge}{\Gamma, A \vdash B} \text{ } \vee \wedge \quad \frac{}{\Gamma, B \vdash B} \text{ } \wedge}{\Gamma \vdash (A \vee B) \wedge \neg A \vdash B} \text{ } \vee \quad \frac{}{\vdash ((A \vee B) \wedge \neg A) \rightarrow B} \text{ } \rightarrow$$

③ Comme $\mu(\emptyset) = \mathbb{R}$ il faut montrer que

$$\mu(((A \vee B) \wedge \neg A) \rightarrow B) = \mathbb{R}$$

on a: $\mu(A \vee B) = \mu(A) \cup \mu(B)$

$$\text{et } \mu((A \vee B) \wedge \neg A) = (\mu(A) \cup \mu(B)) \cap \overline{\mu(A)^c}$$

$$\text{or } \overline{\mu(A)^c} \subset \mu(A)^c \quad \text{donc:}$$

$$\begin{aligned} \mu((A \vee B) \wedge \neg A) &= \emptyset \cup \mu(B) \cap \overline{\mu(A)^c} \\ &= \mu(B) \cap \overline{\mu(A)^c} \end{aligned}$$

et ainsi,

$$\mu(((A \vee B) \wedge \neg A) \rightarrow B) = \overline{(\mu(B) \cap \overline{\mu(A)^c})^c} \cup \mu(B)$$

$$\begin{aligned} \text{or } (\mu(B) \cap \overline{\mu(A)^c})^c &= \mu(B)^c \cup \overline{(\mu(A)^c)^c} \\ &\supset \mu(B)^c \end{aligned}$$

~~Donc~~ donc

$$\mu(((A \vee B) \wedge \neg A) \rightarrow B) \supset \mu(B) \cup \mu(B)^c = \mathbb{R}$$

$$\frac{\frac{\frac{A \rightarrow B \vdash A \vee \neg A}{\vdash A \rightarrow B} \text{TE} \quad ; \quad \frac{\frac{\frac{\frac{\frac{\frac{A \rightarrow B, A \vdash A}{\vdash A} \text{pr} \quad ; \quad \frac{\frac{A \rightarrow B, A \vdash A \rightarrow B}{\vdash A \rightarrow B} \text{pr}}{\vdash A \rightarrow B, A \vdash B} \text{pr}}{\vdash A \rightarrow B, A \vdash \neg A \vee B} \text{iv} \quad ; \quad \frac{\frac{A \rightarrow B, \neg A \vdash \neg A}{\vdash A \rightarrow B, \neg A \vdash \neg A} \text{pr}}{\vdash A \rightarrow B, \neg A \vdash \neg A \vee B} \text{iv}}{\vdash A \rightarrow B, A \vdash \neg A \vee B} \text{ev}}{\vdash A \rightarrow B \vdash \neg A \vee B} \text{iv} \rightarrow \quad \vdash (A \rightarrow B) \rightarrow (\neg A \vee B)$$
$$\mu(A \rightarrow B) = \overline{\mu(A)^c \cap \mu(B)} = \overset{0}{1}R = 1R$$

$$\mu(\neg A \vee B) = \overbrace{\mu(A)^c}^{\emptyset} \cup \mu(B) = \emptyset \cup IR^* = IR^*$$

gnc:

$$\mu((A \rightarrow B) \rightarrow (\neg A \vee B)) = \frac{0}{\emptyset \cup \mathbb{R}^*} = \mathbb{R}^* \neq \mathbb{R}$$

Ce séquent n'est donc pas valide pour la sémantique de Heyting.

⑥

$$* \frac{\Gamma \vdash A ; \Gamma \vdash A \rightarrow B}{\Gamma \vdash B} \rightarrow e$$

on suppose que $\mu(\Gamma) \subset \mu(A)$ et $\mu(\Gamma) \subset \mu(A \rightarrow B) = \overline{\mu(A)^c \cup \mu(B)}$

Ainsi :

$$\mu(\Gamma) \subset \mu(A) \cap (\overline{\mu(A)^c \cup \mu(B)}) \Rightarrow \mu(A) \cap (\overline{\mu(A)^c \cup \mu(B)}) = \overline{\mu(A) \cap \mu(B)} \subset \mu(B)$$

car $\mu(A) = \mu(A) \circ$ (ouvert)
et $U \cap V = \overline{U \cap V}$

*

⑦ Induction sur l'arbre de preuve

⑧ On pose $\mu(A) = \mathbb{R}^*$ alors

$$\mu(A \vee \neg A) = \mu(A) \cup \overline{\mu(A)^c} = \mathbb{R}^* \neq \mathbb{R}.$$

Ceci nous donne une formule qui n'est pas valide pour la sémantique de Heyting. Le système intuitionniste est correct pour cette sémantique d'après la question 3 donc cette formule ne peut pas être prouvée en logique intuitionniste car cette correction garantirait alors sa validité - absurde.

⑨

```
int sl-recherche(slliste* l, char* def) {
```

```
    slNoeud* curr = l->sentinelle; → suivant-par-niveau [niveau]
```

```
    slNoeud* save = l->sentinelle;
```

```
    int niveau = l->niveau-actuel;
```

```
    while (niveau > 0) {
```

```
        while (curr != NULL && strcmp(curr->def, def) < 0)
```

```
        { save = curr;
```

```
          curr = curr->suivant-par-niveau[niveau];
```

```
        }
```

```
        if (strcmp(curr->def, def) == 0)
```

```
        if (curr == NULL) {
```

```
            niveau --;
```

```
            curr = save;
```

```
        }
```

```
        else if (strcmp(curr->def, def) == 0) {
```

```
            return (curr->val);
```

```
        }
```

```
        else { niveau --;
```

```
              curr = save;
```

```
        }
```

```
    }
```

```
    return (-1);
```

```
}
```

⑩ Si $k = \text{MAX_NIVEAU}$, la proba que la taille soit k est $\frac{1}{k}$
et pour $k \in [0, \text{MAX_NIVEAU} - 1]$ c'est $\frac{1}{2^{k+1}}$.

⑪

```
for (int n = l → niveau-actuel; n >= 0; n--) {  
    /* on fait un parcours depuis tt en haut pour plus  
    d'efficacité de l'accès ciblé à un niveau donné */  
    sl_Noed* suivant = courant → suivant-par-niveau[n];  
    while (suivant != NULL && strcmp(suivant → de, de) < 0) {  
        courant = suivant;  
        suivant = suivant → suivant-par-niveau[n];  
    }  
    noeuds-a-mettre-a-jour[n] = courant;  
}
```

⑫

ajouter
boucle →

```
liste → niveau-actuel = niveau;  
liste → sentinelle → suivant-par-niveau[niveau]  
= nouveau-noeud;
```

il faut le faire pour chaque niveau de niveau-actuel à niveau

⑬

```
for (int n = niveau; n >= 0; n--) {  
    sl_Noed* no = noeuds-a-mettre-a-jour[n];  
    sl_Noed* next = no → suivant-par-niveau[n];  
    no → suivant-par-niveau[n] = nouveau-noeud;  
    nouveau-noeud → suivant-par-niveau[n] = next;  
}
```

(14) Las Vegas.

(15) Cela revient à faire une dichotomie: à chaque niveau on fait une comparaison qui découpe l'ensemble des valeurs possibles en 2 et donc la complexité est en $O(\log_2(n))$.

(16) Soit N_k la va qui représente le nb d'élts au niveau k et n le nb d'élts de la skip-list.

$$E(N_0) = n \text{ et}$$

$$\forall k \geq 0, \quad E(N_{k+1}) = \frac{1}{2} E(N_k)$$

$$\text{Ainsi, } \forall k \geq 0, \quad E(N_k) = \frac{n}{2^k} \quad (\text{recours } k)$$

(17) Soit Π_n le niveau maximum atteint.

$$E(\Pi_n) = \sum_{k=0}^{+\infty} k \Pr(\Pi_n = k) = \sum_{k=1}^{+\infty} \Pr(\Pi_n \geq k)$$

$$\begin{aligned} \text{avec } \Pr(\Pi_n \geq k) &= 1 - \Pr(\Pi_n < k) \\ &= 1 - \left(1 - \frac{1}{2^k}\right)^n \end{aligned}$$

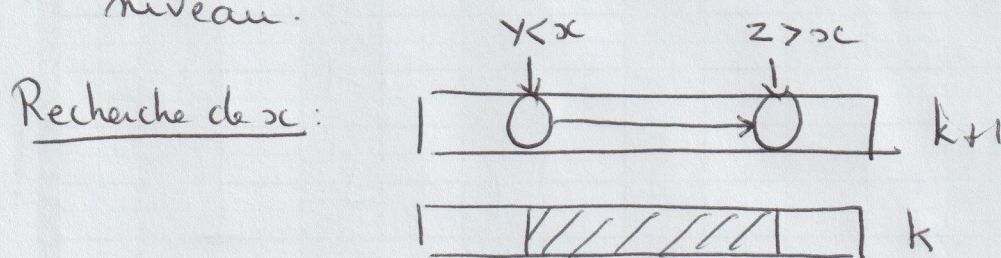
en effet: $\Pr(\text{un nœud soit de hauteur au } \geq k \text{ vœut}) = \frac{1}{2^k}$

On obtient donc:

$$\underline{E(\Pi_n) = \sum_{k=1}^{+\infty} \left(1 - \left(1 - \frac{1}{2^k}\right)^n\right)}$$

18

Considérons une recherche dans le niveau k . Elle s'arrêtera au pire dès lors qu'on rencontrera un noeud qui est aussi présent au niveau $k+1$ car l'on parcourt les sommets du niveau k compris entre les deux sommets du niveau $k+1$ qui nous ont fait descendre de niveau.



Chaque sommet du niveau k a une chance sur 2 d'être au niveau $k+1$ donc le nombre de noeuds parcourus au niveau k suit une loi géométrique de paramètre $\frac{1}{2}$ dont l'espérance est 2.

On parcourt donc 2 noeuds horizontalement en moyenne

19 On a une hauteur moyenne de la liste qui est logarithmique par rapport à n et un nombre moyen de comparaisons par étage qui vaut 2

Ainsi, l'espérance de la recherche est en $O(\log(n))$

(20)

$$C' \rightarrow C' L \mid L$$

(21)

$$V \rightarrow M V' \mid N \quad \text{et} \quad V' \rightarrow N V' \mid \epsilon$$

(22)

$$T \rightarrow \{ S \}$$

$$\rightarrow \{ K, S \} \rightarrow \{ K, K, S \} \rightarrow \{ K, K, K \}$$

$$\rightarrow \{ C:V, K, K \}$$

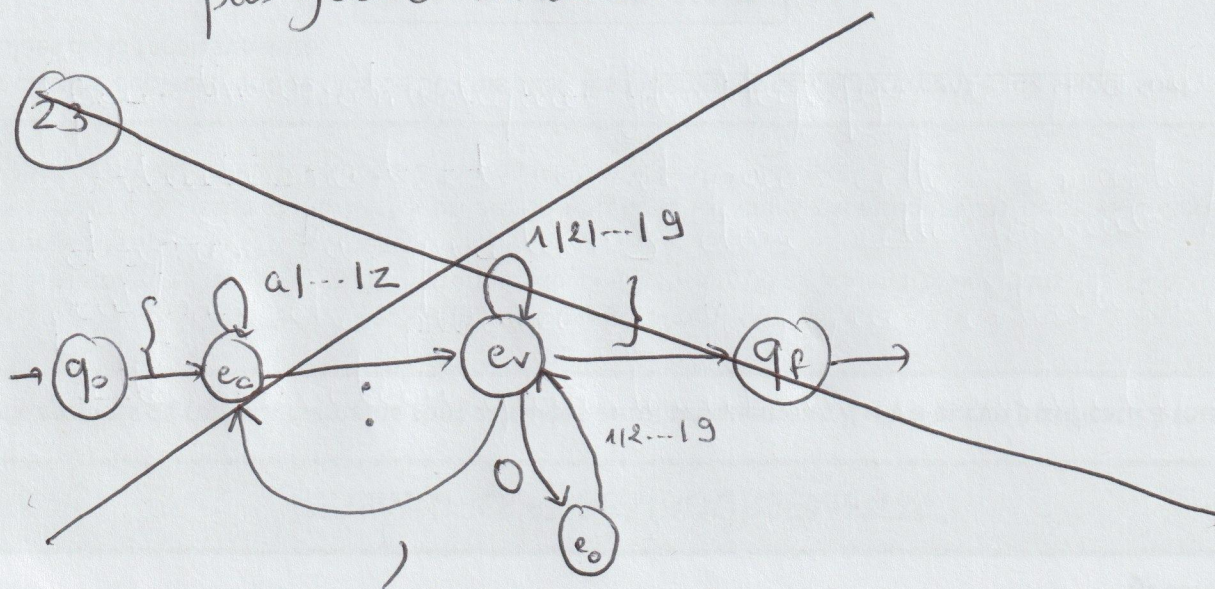
$$\rightarrow \{ C:V, C:V, K \}$$

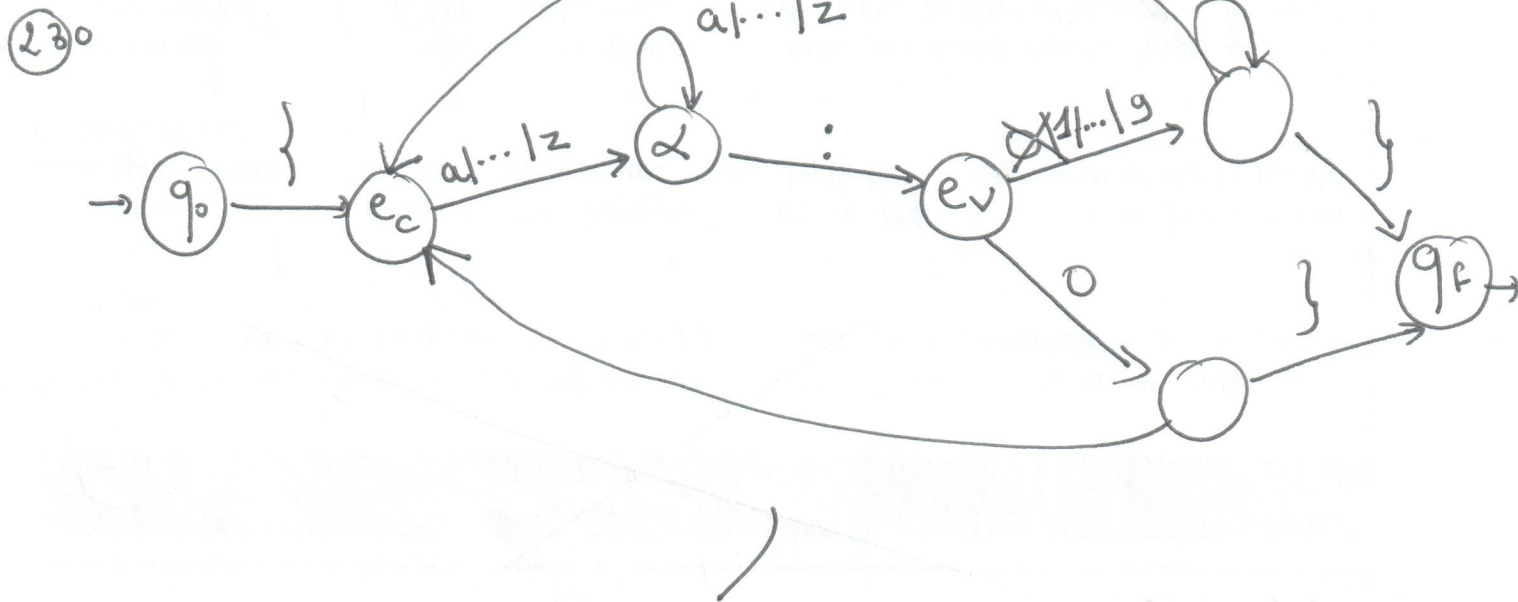
$$\rightarrow \{ C:V, C:V, C:V \}$$

et à partir de C et de V on engendrera
id, valeur, ok, 5, 12 et 0 donc le mot est
bien généré par la grammaire

Le deuxième mot ne l'est pas car on ne saura
pas générer "la valeur" "007".

(23)





$$(25) \begin{pmatrix} bba \\ bb \end{pmatrix} \begin{pmatrix} ab \\ aa \end{pmatrix} \begin{pmatrix} bba \\ bb \end{pmatrix} \begin{pmatrix} a \\ paa \end{pmatrix}$$

3 2 3 1 fournit une solut; c'est bien une instance positive du pb.

(26) La combinaison commence forcément par

$$\begin{pmatrix} abc \\ ab \end{pmatrix}$$

1

car c'est le seul domino dont les lettres en haut et en bas coïncident.

Aucun domino n'a de motif du bas qui commence par c, on est donc bloqué et l'instance est négative.

(27) Si on a une solut^o $(i_1, \dots, i_k)$ alors pour tout entier c, on a $(i_1, \dots, i_k, \underbrace{i_1, \dots, i_k}_{\text{répété c fois}} \dots)$

qui est une autre solution et donc on peut construire une infinité de solut^o à partir de l'une d'entre elles.

(28) cours : Si on a A_2 un algo qui décide f_2 alors si $g: E_1 \rightarrow E_2$ est la réduction supposée par l'énoncé, on aurait A_1 sur entrée $x \in E_1$

renvoyer $A_2(g(x))$

qui décide f_1 ce qui est absurde car supposé impossible

29

$$S \rightarrow u_i S a_i \text{ pour chaque } i \in [1, n]$$

$$S \rightarrow \varepsilon u_i a_i$$

engendre L_u .

30

Si PCP est positive alors $L_u \cap L_v \neq \emptyset$

Réciproquement, si $L_u \cap L_v \neq \emptyset$ alors m est engendré à la fois par une grammaire pour L_u et une pour L_v et alors $m = u_{i_1} \dots u_{i_k} a_{i_1} \dots a_{i_k}$

$$= \underbrace{v_{i_1} \dots v_{i_k}}_{\text{les } \tilde{m} \text{ indices}} \overbrace{a_{i_1} \dots a_{i_k}}^{\text{les } \tilde{m} \text{ lettres}}$$

$(i_1, \dots, i_k)$ fournit alors une solution au problème PCP.

31 L'équivalence de la q° 30 garantit que PCP $\leq$ inter-vide

$$\begin{pmatrix} u_1 \\ \vdots \\ v_1 \end{pmatrix} \dots \begin{pmatrix} u_n \\ \vdots \\ v_n \end{pmatrix} \longrightarrow L_u, L_v$$

32 Cette grammaire engendre $L_u \cup L_v$

33 Réducto: $\begin{pmatrix} u_1 \\ \vdots \\ v_1 \end{pmatrix} \dots \begin{pmatrix} u_n \\ \vdots \\ v_n \end{pmatrix} \longrightarrow \text{grammaire de la q° 32}$

si $\begin{pmatrix} u_1 \\ \vdots \\ a_{i_1} \end{pmatrix} \dots \begin{pmatrix} u_n \\ \vdots \\ a_{i_n} \end{pmatrix}$ est une instance ≥ 0 de PCP alors la grammaire associée est ambiguë car si $(i_1, \dots, i_k)$ est une solution, on aura $u_{i_1} \dots u_{i_k} a_{i_k} \dots a_{i_1}$ et $v_{i_1} \dots v_{i_k} a_{i_k} \dots a_{i_1}$ qui sont égaux et qui correspondent à deux arbres $\neq$: l'un commençant par $S \rightarrow U$ et l'autre par $S \rightarrow V$

Réciproquement, si la grammaire est ambiguë, en remarquant que les mots engendrés depuis U (resp V) le sont de manière non ambiguë alors l'ambiguïté vient forcément d'un mot de $L_u \cap L_v$ qui est engendré d'une part depuis U et d'autre part depuis V et ainsi le mot $u_{i_1} \dots u_{i_k} a_{i_k} \dots a_{i_1}$ ainsi engendré fournit $(i_1, \dots, i_k)$ comme solution à l'instance de PCP

(34) C'est le complémentaire de L_u .

Réduct

(35)

$$S \rightarrow D \beta D \alpha D$$

$$D \rightarrow \alpha D \mid \beta D \mid \varepsilon$$

pour chaque $\alpha \in \Sigma$
et $\beta \in \Delta$

(36)

~~est une~~

$$S \rightarrow u_i S a_i \mid C \text{ pour chaque } i \in \llbracket 1, m \rrbracket$$

$$C \rightarrow u_i A a_j \text{ pour chaque } (i, j) \in \llbracket 1, m \rrbracket^2 \\ i \neq j$$

$$A \rightarrow \varepsilon \mid \underbrace{\alpha A \beta}_{\text{inutile}} \mid \alpha A \mid A \beta \text{ avec } \alpha \in \Sigma \text{ et } \beta \in \Delta$$

(37)

On a vu que $\begin{pmatrix} u_1 \\ \vdots \\ u_m \end{pmatrix} \dots \begin{pmatrix} u_n \\ \vdots \\ u_m \end{pmatrix}$ est une
instance positive de PCP ssi $L_u \cap L_v \neq \emptyset$
i.e ssi ${}^c L_u \cup {}^c L_v \neq \Sigma^{**}$

Or on vient de voir comment construire
une grammaire pour ${}^c L_u$ en faisant une
grammaire qui engendre $L'_u \cup (\Sigma^* \Delta^*) \cup \{\varepsilon\}$
ce qui est faisable d'après les q^o 35 et 36
et en utilisant la stabilité par union.
La stabilité par union permet aussi
d'obtenir une grammaire pour ${}^c L_u \cup {}^c L_v$ qui
sera universelle ssi l'instance de PCP est négative

(38)

Soit $S \rightarrow \alpha S | \epsilon \quad \forall \alpha \in \Sigma$

alors ~~le~~ le langage engendré est Σ^*

Ainsi

$Univ \leq Egal$

$\mathcal{G} \rightarrow (\mathcal{G}, \mathcal{G}_u)$ où $\mathcal{G}_u$ est la

grammaire définie ci-dessus engendrant Σ^* .

On a bien que $\mathcal{G} \in Univ$ ssi $(\mathcal{G}, \mathcal{G}_u) \in Egal$

Ainsi, $Univ \leq Egal$ et $Univ$ est indécidable
car PCP l'est d'après la q^o 37 donc $Egal$ est aussi
indécidable.