Approche par contraintes pour une classe d'emplois du temps universitaires

Université d'été de Le Mans Université, 2024

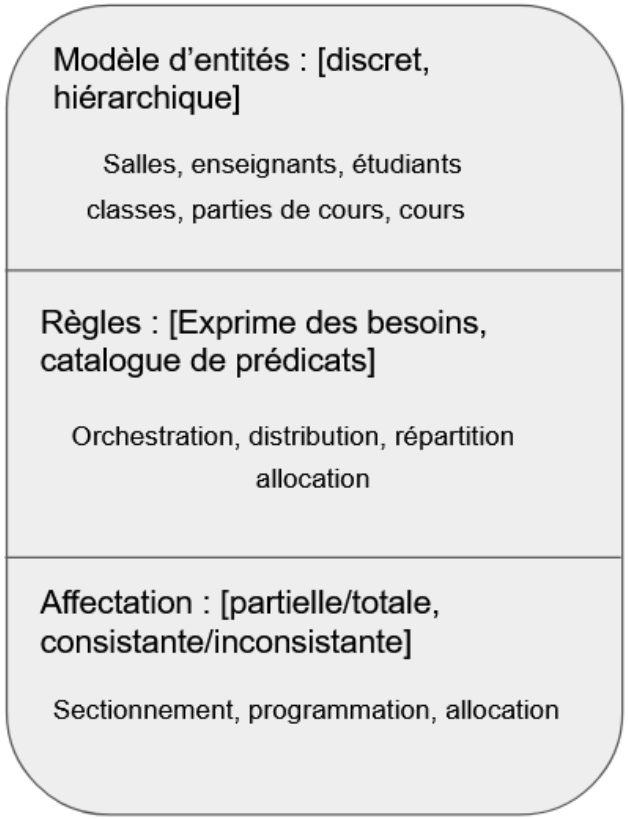
Équipe : V. Barichard, C. Behuet, D. Genest, M. Legeay, D. Lesaint

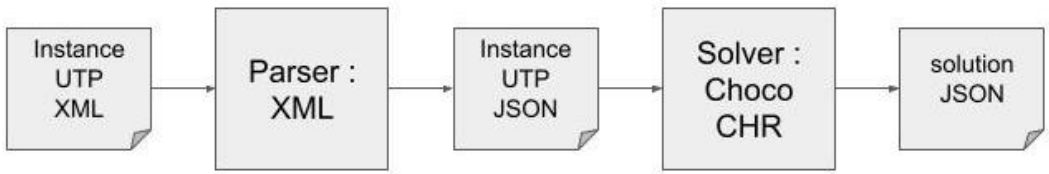
Université d'Angers - LERIA
https://ua-usp.github.io/timetabling/

Le Mans | Jeudi 20 Juin 2024

Langage UTP : description

- UTP (University Timetable Problem), une nouvelle classe de problèmes : planification des cours, allocation des ressources et répartition des étudiants
- Langage UTP : un langage dédié écrit en XML
- Chaîne de traitement :





Exemple de XML

Ressources example xml

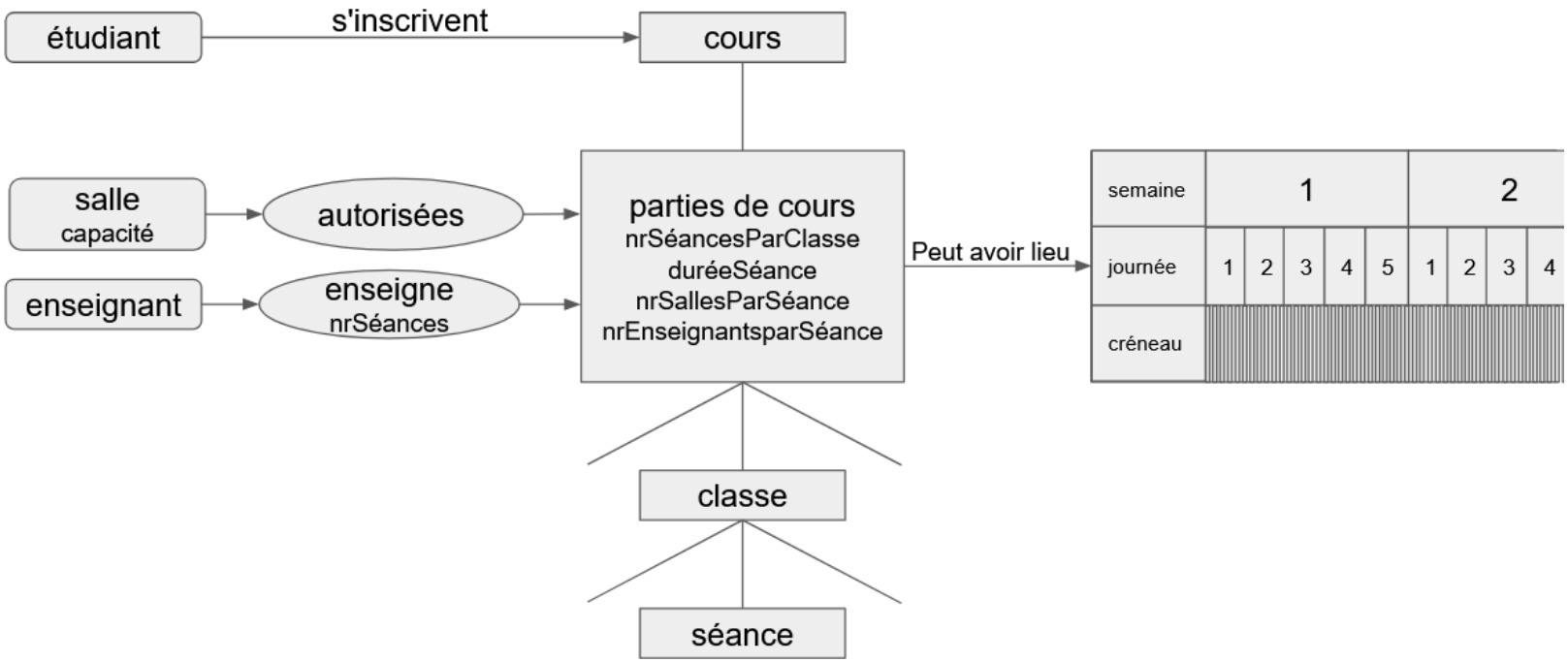
```xml
<?xml version="1.0" encoding="UTF-8"?>
    <rooms>
    <room id="AMPHI-A" capacity="90" label="BATIMENT-A"/>
    <room id="AMPHI-B" capacity="286" label="BATIMENT-A"/>
    <room id="H001" capacity="20" label="BATIMENT-H"/>
    <room id="H002" capacity="42" label="BATIMENT-H"/>
    <room id="H003" capacity="38" label="BATIMENT-H"/>
    </rooms>

    <teachers>
        <teacher id="LESAINT David" label="DEPARTEMENT-
            INFORMATIQUE"/>
        <teacher id="JAMIN Antoine" label="DEPARTEMENT-
            INFORMATIQUE"/>
    </teachers>

    <students>
        <student id="S1" label="Student564">
            <courses>
                <course refId="cours-1"/>
                <course refId="cours-2"/>
            </courses>
        </student>
        <student id="S2" label="Student565">
            <courses>
                <course refId="cours-1"/>
                <course refId="cours-2"/>
            </courses>
        </student>
    </students>
```

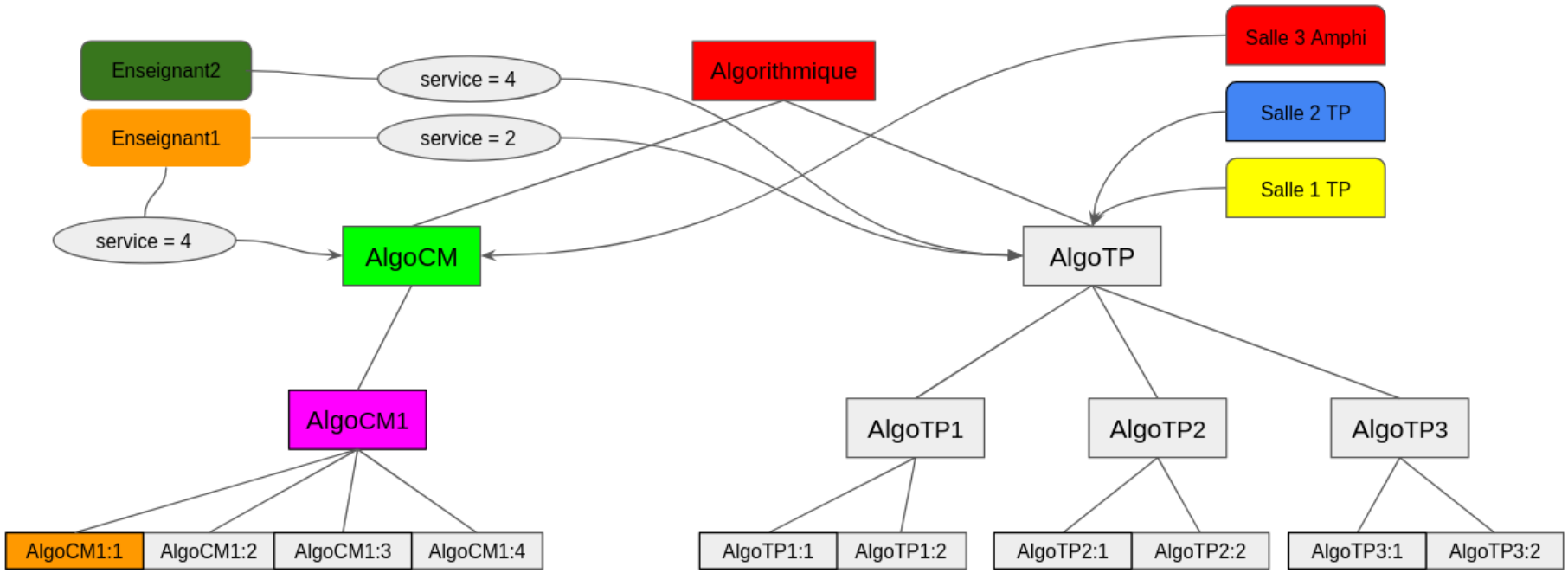
Langage UTP : modèle d'entité

- Horizon de temps à 3 niveaux
- Entités : éléments de maquette de formations, ressources
- Contraintes de domaine et de distribution des ressources



- Séances ordonnées par classe

Exemple


Enseignant2
Enseignant1
service = 4
service = 2
service = 4
Algorithmique
Salle 3 Amphi
Salle 2 TP
Salle 1 TP
AlgoCM
AlgoTP
AlgoCM1
AlgoTP1
AlgoTP2
AlgoTP3
AlgoCM1:1
AlgoCM1:2
AlgoCM1:3
AlgoCM1:4
AlgoTP1:1
AlgoTP1:2
AlgoTP2:1
AlgoTP2:2
AlgoTP3:1
AlgoTP3:2

Contraintes

- Certaines contraintes sont intégrées dans le modèle d'entité
  - Pré-allocation de ressources (enseignants, salles, étudiants)
  - Ordre des séances au sein d'une classe
- Définition des prédicats `UTP`
  - *sequenced*
  - *same_rooms*
  - *forbidden_period*
  - ...
- Les contraintes s'appliquent à des ensembles de séances liées à une ressource (**e-maps**)

$$\underbrace{predicate}_{\text{symbole de prédicat}} \left(\underbrace{(e_1, S_1), \ldots, (e_m, S_m)}_{\text{ensemble des couples (entité, séances)}} , \underbrace{p_1, \ldots, p_n}_{\text{ensemble des paramètres}} \right)$$

- Les contraintes peuvent avoir des paramètres
- Les contraintes sont conditionnelles :
  $\forall i \in 1..m,\ \forall T \subseteq S_i : x_e^{(S_i)} = T \rightarrow c(x_e^{(S_i)}, S_i) \leftrightarrow c(x_e^{(S_i)}, T \cap S_i)$

Règles

- Les règles génère des collections de contraintes
- Langage de requête similaire à sql ou x-path

$$\Big(\underbrace{G_1{:}F_1, \ldots, G_m{:}F_m}_{\text{ensemble des couples (générateur, filtre)}} , \underbrace{predicate}_{\text{symbole de prédicat}} , \underbrace{p_1, \ldots, p_n}_{\text{ensemble des paramètres}} \Big)$$

- Générateur : il regroupe les séances selon le **type** indiqué (class, part, enseignant ...) et un **masque** de séances (séances 1 à 5). Un générateur est un n-uplet sous la forme :

$$\forall i \in 1..m,\ G_i = (type, masque)$$

- Filtre : récupère seulement les séances visées, selon le **type** indiqué et un **identifiant** (ou des étiquettes). Il se présente sous la forme d'une combinaison de n-uplet :

$$\forall i \in 1..m,\ F_i = (type_1, identifiant_1) \wedge (type_2, identifiant_2)$$

Exemple de XML

Ressources example xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
       <rules>
           <rule>
               <selector generator="(class, *)" filters="
                   course[id='cours-1']"/>
               <constraint name="sameRooms" type="hard"/>
           </rule>
           <rule>
               <selector generator="(class, *)" filters="part
                   [id='cours-1-pTP']"/>
               <constraint name="sameTeachers" type="hard"/>
           </rule>
           <rule>
               <selector generator="(class, {4})" filters="
                   course[id='cours-1'],part[label='CM']"/>
               <selector generator="(class, {1})" filters="
                   course[id='cours-1'],part[label='TP']"/>
               <constraint name="sequenced" type="hard"/>
           </rule>
           <rule>
               <selector generator="(class, *)" filters="
                   course[id='cours-1'],part[label='CM']"/>
               <constraint name="periodic" type="hard">
                   <parameters>
                       <parameter name="value">1</parameter>
                       <parameter name="unit">week</parameter
                           >
                   </parameters>
               </constraint>
           </rule>
       </rules>
```

Exemple de règle 1

- La première séance de TP commence après la troisième séance de CM.

$$(((class, \{3\}):(part, algoCM)), (class, \{1\}:(part, algoTP)), sequenced)$$

Exemple de règle 1

- La première séance de TP commence après la troisième séance de CM.

$$(((class, \{3\}):(part, algoCM)), (class, \{1\}:(part, algoTP)), sequenced)$$

- Donnant les contraintes :

$$sequenced((algoCM1, algoCM1{:}3), (algoTP1, algoTP1{:}1))$$

$$sequenced((algoCM1, algoCM1{:}3), (algoTP2, algoTP2{:}1))$$

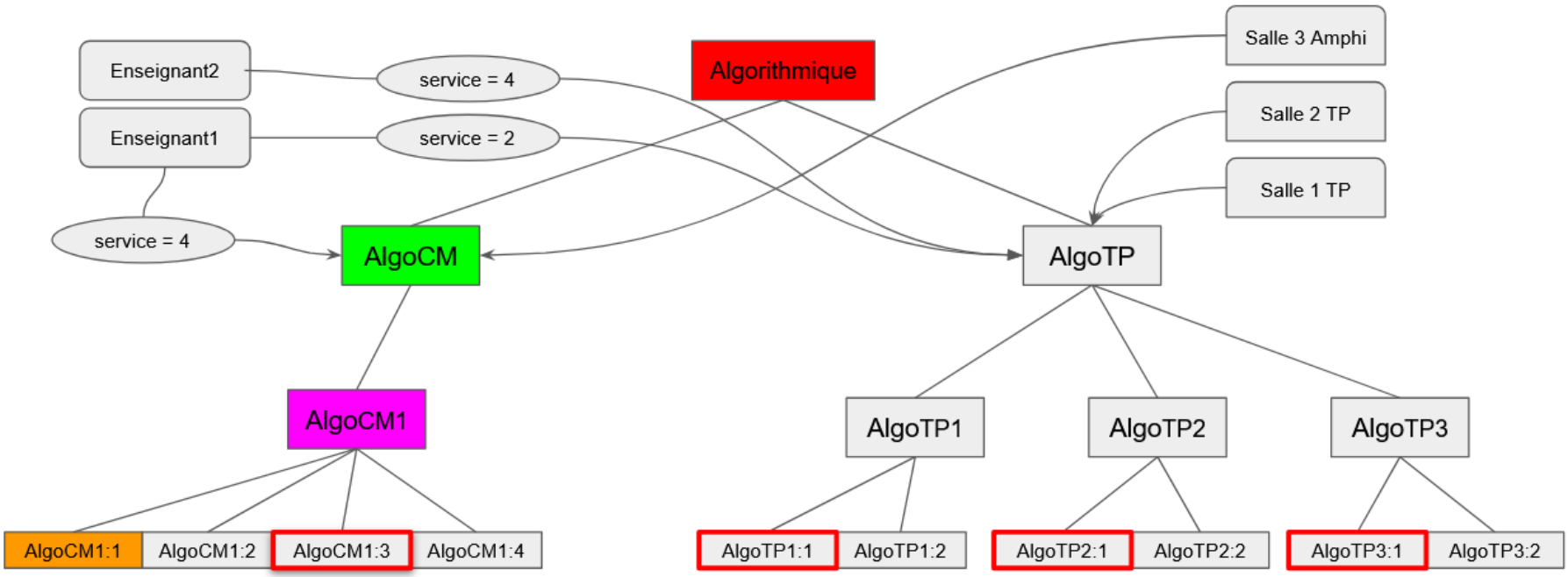
$$sequenced((algoCM1, algoCM1{:}3), (algoTP3, algoTP3{:}1))$$

Exemple de XML

Ressources example xml

```
<?xml version="1.0" encoding="UTF-8"?>
            <rule>
                <selector generator="(class, {3})" filters="
                    course[id='Algorithmique'],part[label='CM
                    ']"/>
                <selector generator="(class, {1})" filters="
                    course[id='Algorithmique'],part[label='TP
                    ']"/>
                <constraint name="sequenced" type="hard"/>
            </rule>
```

Exemple de règle 1



Besoin

La première séance de TP commence après la troisième séance de CM.

Règle

$(((class, \{3\}):(part, algoCM)), (class, \{1\}:(part, algoTP)), sequenced)$

Contraintes

$sequenced((algoCM1, algoCM1{:}3), (algoTP2, algoTP1{:}1))$

$sequenced((algoCM1, algoCM1{:}3), (algoTP2, algoTP2{:}1))$

$sequenced((algoCM1, algoCM1{:}3), (algoTP2, algoTP3{:}1))$

Exemple de règle 2

- Les séances de TP de l'enseignant 2 ne peuvent pas avoir lieu le mardi de la 2ème semaine entre 8h et 10h.

  $((teacher, \{*\}):(part, algoTP), (teacher, enseignant_2)),$
  $forbidden_period, 9120, 9240)$

Exemple de règle 2

- Les séances de TP de l'enseignant 2 ne peuvent pas avoir lieu le mardi de la 2ème semaine entre 8h et 10h.

  $((teacher, \{*\}):(part, algoTP), (teacher, enseignant_2)),$
  $forbidden_period, 9120, 9240)$

- Donnant les contraintes :

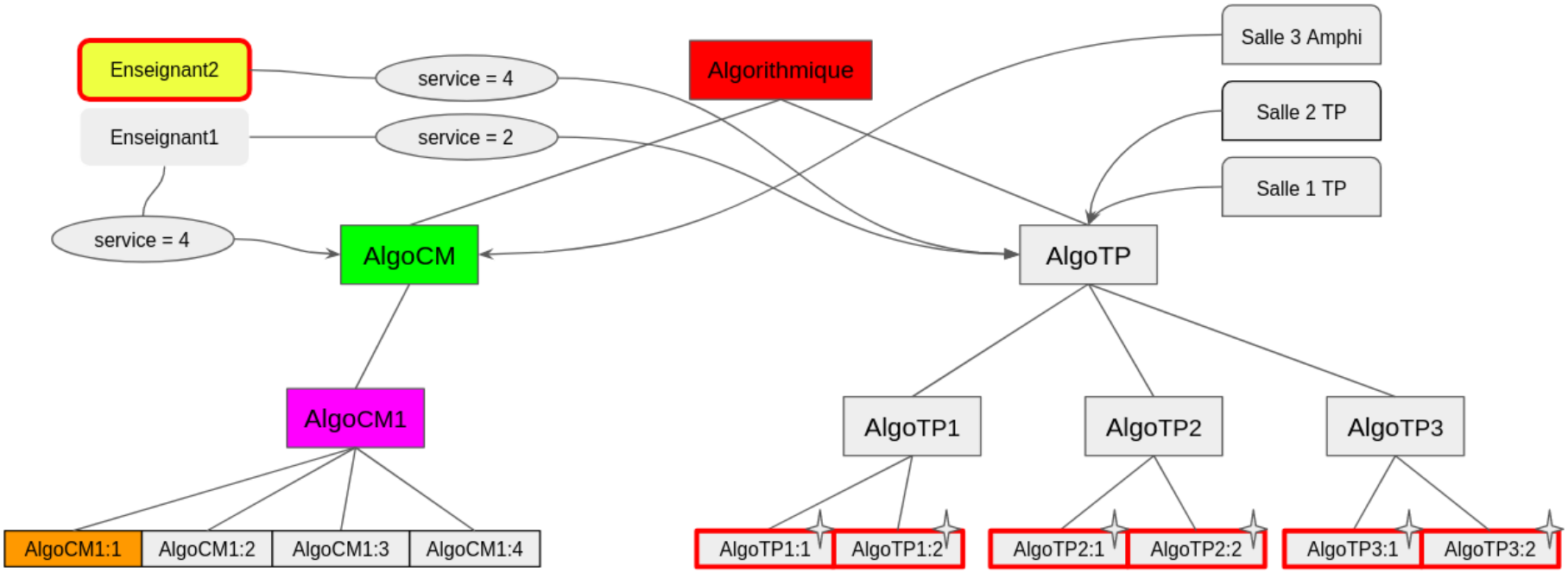
  $forbidden_period(enseignant2, (algoTP1{:}1, algoTP1{:}2, algoTP2{:}1,$
  $algoTP2{:}2, algoTP3{:}1, algoTP3{:}2)), 9120, 9240)$

Exemple de XML

Ressources example xml

```
<?xml version="1.0" encoding="UTF-8"?>
            <rule>
                <selector generator="(teacher[id='enseignant2
                    '], {*})" filters="course[id='
                    Algorithmique'],part[label='TP']"/>
                <constraint name="forbiddenPeriod" type="hard"
                    />
            </rule>
```

Exemple de règle 2



Besoin

Les séances de TP de l'enseignant 2 ne peuvent pas avoir lieu le mardi de la 2ème semaine entre 8h et 10h.

Règle

$((teacher, \{*\}):(part, algoTP), (teacher, enseignant_2)), forbidden_period, 9120, 9240)$

Contraintes

$forbidden_period(enseignant2, (algoTP1{:}1, algoTP1{:}2, algoTP2{:}1, algoTP2{:}2, algoTP3{:}1, algoTP3{:}2), 9120, 9240)$

✦ : L'étoile indique la conditionnalité

Résolution
programmation par contraintes

Nous avons défini un modèle en programmation par contraintes (CHOCO) :

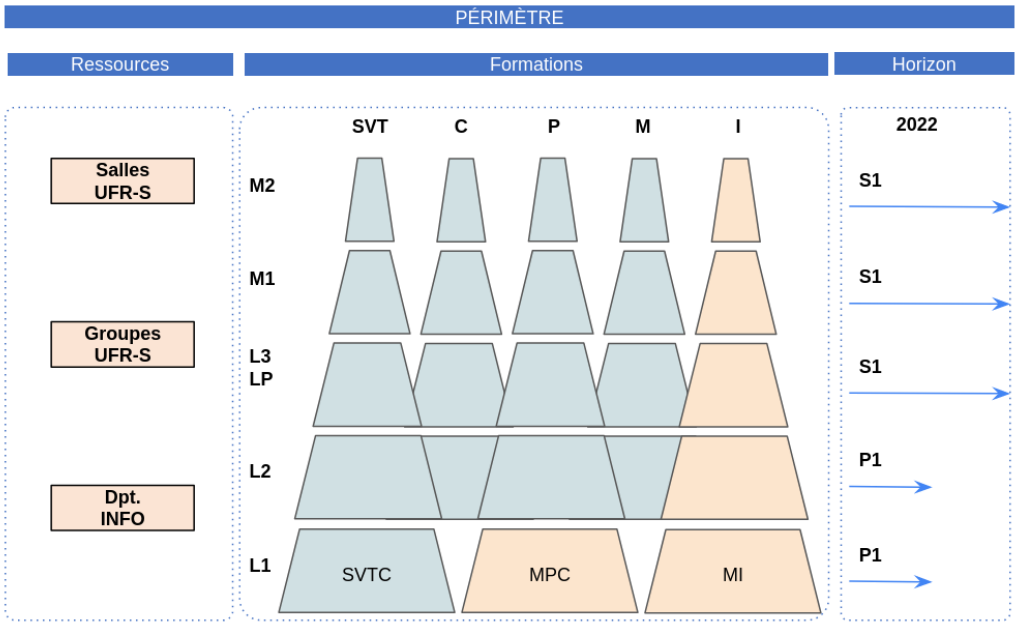
- Variables de décision :
  - Variable ensemblistes (choix des enseignants, salles)
  - Variables entières (choix du créneau de début)
- Contraintes globales (global cardinality, cumulatives, all_equal...)
- Reformulation des contraintes pour des instances spécifiques
- Stratégie :
  - Salles, enseignants, puis créneaux
  - Choix aléatoire pour les salles et les enseignants
  - Partage du domaine pour les créneaux

Génération

- Générateur d'instances
  - Tester la robustesse et l'efficacité des différentes approches
  - Identifier des familles d'instances
  - Réaliser des simulations au vue des données d'entrée
- Difficulté
  - Identifier les liens au sein d'une instance
  - Créer de manière pseudo aléatoire des instances
  - Résoudre les les sous-problèmes tel que le service ou les groupes d'étudiant
- Perspectives
  - Génération via des méthodes issues de l'apprentissage
  - Identifier le lien entre les caractéristiques des instances et sa difficulté
  - Choix des paramètres en fonction de la famille d'instances

Expérimentation : instance réelle

- Instance L1,L2 + L3,M1,M2 Informatique de l'université d'Angers :
  - 24 semaines, 5 jours, 1440 créneaux quotidiens
  - 370 classes, 2500 séances
  - 140 salles, 100 enseignants, 800 étudiants, 60 groupes
- Taille du modèle :
  - 95 000 variables
  - 79 000 contraintes
  - 20s de résolution
  - 1500 fail



Outillage & perspectives

- Outils
  - Langage UTP (catalogue de contrainte)
  - 2 approches par contraintes (Choco, CHR++)
  - Plusieurs approches en résolution exacte(ASP, CASP et MIP)
  - Visualisation performantes des EDT
  - Un générateur pseudo-booléen
  - Une chaîne de traitement (parser...)
- En développement
  - Interface pour drag and drop
  - Interface de saisie de l'EDT
  - Algorithme d'aide à la réparation
  - Interface de saisie des règles avec LLM
  - Interface pour les services des enseignants