Zoé Marquis & Charlotte Kruzic

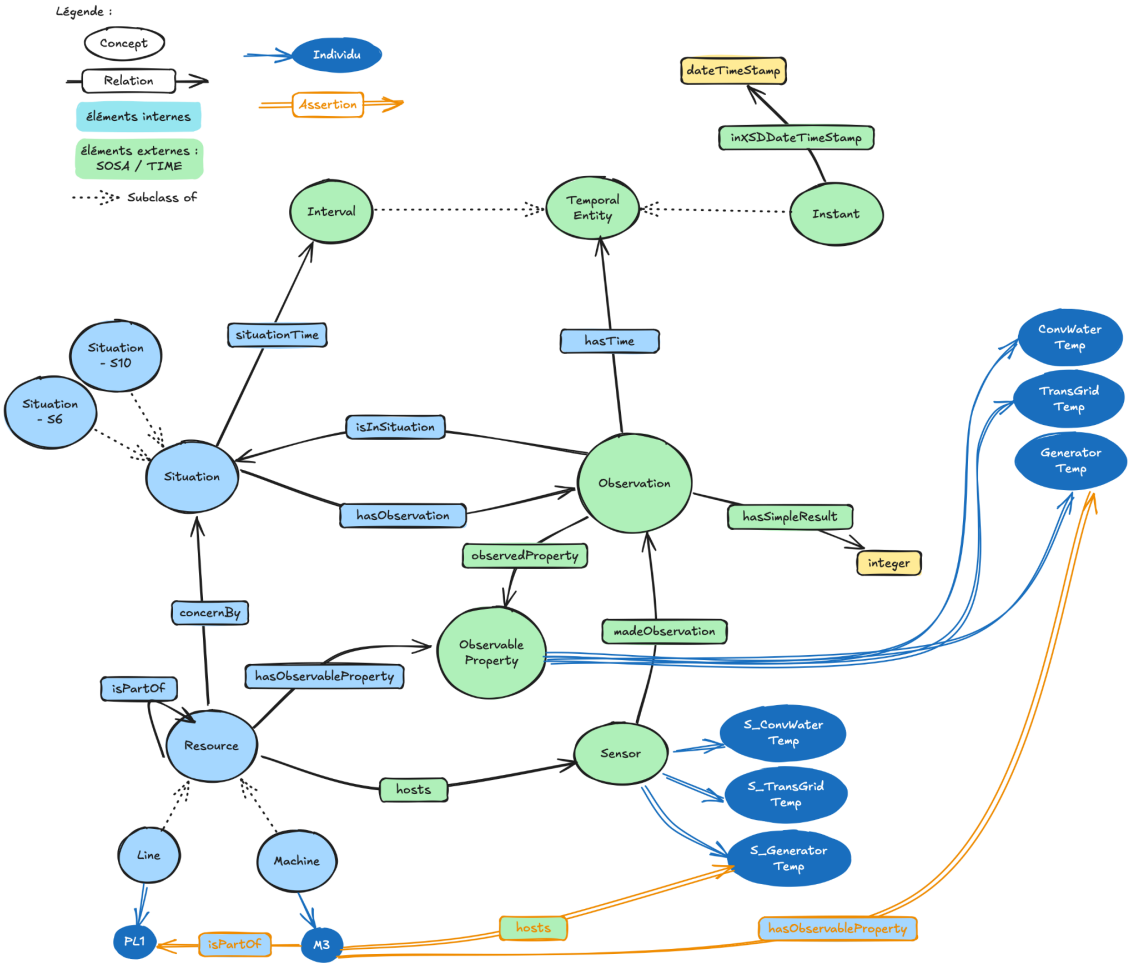
Apprentissage Collaboratif
Combining knowledge-based with data-driven approaches

1. Introduction briefly explaining the application scenario.

Objectif
Le projet vise à appliquer le raisonnement sur flux de données (stream reasoning) pour détecter des situations anormales dans un contexte industriel. Ces données sont exploitées pour surveiller la production et identifier rapidement des défaillances potentielles.
Scénario industriel
Le cas étudié porte sur une ligne de production (PL1) composée de quatre machines : M1, M2, M3, et M4. Chaque machine est équipée de capteurs (*Sensors*) mesurant différentes propriétés, selon leur rôle. Toutes les machines ne disposent donc pas des mêmes types de capteurs.
Défaillances étudiées
Les défaillances sont définies sous forme de contraintes à vérifier. Étant donné les contraintes temporelles du projet, nous avons focalisé l'analyse sur deux situations spécifiques (S6 et S10).

2. Description of the developed ontology.

Nous avons choisi une représentation sous forme de schéma pour sa clarté et sa capacité à illustrer de manière visuelle les concepts, relations et instances.
Pour préserver la lisibilité, seuls les individus et assertions nécessaires à la détection de la situation 6 ont été inclus, afin d'éviter une surcharge d'informations.
Le schéma présenté correspond à l'état de l'ontologie avant l'application du stream reasoning et l'insertion d'individus détectés pour les différentes situations.



3. Description of the RDF streams generated as well as the required C-SPARQL queries.

RDF quadruplets observés pour l'observation numéro y :

```
(S_x, madeObservation, S_x-Obs-y) . (TIMESTAMP)
(S_x-Obs-y, observedProperty, x) . (TIMESTAMP)
(S_x-Obs-y, hasSimpleResult, INTEGER) . (TIMESTAMP)
(S_x-Obs-y, hasTime, t-obs-S_x-y) . (TIMESTAMP)
(t-obs-S_x-y inXSDDateTime, DATETIMESTAMP) . (TIMESTAMP)
```

Légende :
`TIMESTAMP` → une valeur de type *timestamp*
`INTEGER` → une valeur de type `XMLSchema#integer`
`DATETIMESTAMP` -> une valeur de type `XMLSchema#dateTimeStamp`

Si on traduit en concept et propriété :
`x` est ici une instance de `ObservableProperty`
`S_x` est une instance de `Sensor`
`S_x-Obs-y` est une instance de `Observation`
`t-obs-S_x-y` est une instance de `TimeInstant` (sous classe de `TemporalEntity`)

```
(Sensor, madeObservation, Observation) . (TIMESTAMP)
(Observation, observedPropert, ObservableProperty) . (TIMESTAMP)
(Observation, hasSimpleResult, INTEGER) . (TIMESTAMP)
(Observation, hasTime, TimeInstant) . (TIMESTAMP)
(TimeInstant inXSDDateTime, DATETIMESTAMP) . (TIMESTAMP)
```

Pour chaque flux de données sur une `ObservableProperty` ces quadruplets RDF sont générés à chaque "moment d'observation".

C-SPARQL query :
Pour ne pas surcharger le rapport vous pouvez retrouver en annexe la requête correspondant à la situation 6.

- **Fenêtre glissante** : On utilise une fenêtre glissante de 15 secondes avec un pas de 5 secondes pour analyser les flux de données ([RANGE 15s STEP 5s]). Pour la situation 10, une fenêtre de 20 secondes est utilisée. Ces tailles ont été déterminées à partir des données fournies dans l'énoncé.
- **Relations de l'ontologie exploitées** :
  - La relation entre les *Machines* et les *Lines* (entre différentes *Resources*) est exploitée (*M3 isPartOf PL1)*
  - La relation *hosts* (définie dans le vocabulaire SOSA) est utilisée pour associer une *Machine* (celle reliée à une autre *Resource* au-dessus) à un *Sensor*.
  - La relation *madeObservation* permet d'établir le lien entre le *Sensor* précédent et l'*Observation* générée par le flux de données.
  - La relation *hasSimpleResult* de l'*Observation* précédente permet de récupérer les valeurs mesurées par les capteurs.
- **Détection des situations par les contraintes** : Les contraintes spécifiques à chaque situation sont intégrées dans le filtre (*FILTER*) de la requête. Cela permet de détecter des combinaisons particulières de valeurs qui correspondent aux situations anormales définies.

4. Analysis of the limitations and how to potentially overcome them.

- **Complexité de mise en œuvre** : Les situations sont basées uniquement sur les connaissances des experts, ce qui peut entraîner des contraintes incomplètes ou biaisées. Solution : Combiner la modélisation par apprentissage automatique (ML) avec les connaissances humaines. *Complex Domains, Incomplete Data*
- **Concept drift** : Les contraintes peuvent évoluer avec le temps, mais il n'y a pas de gestion de ces changements dans le système actuel. Manque de flexibilité pour s'adapter aux évolutions. (Pas de solution à proposer.) *Rapid Evolution*
- **Volumes d'observations importants** : Les données générées par de nombreux capteurs peuvent être difficiles à gérer par le système de traitement du flux de données. Solution : Optimiser la taille des fenêtres et le pas de temps via l'apprentissage automatique. Supprimer les données obsolètes ou en dehors de la fenêtre active. *Vast Data*
- **Rigidité et grand nombre de requêtes** : Le grand nombre de requêtes peut entraîner des délais de réponse trop longs, notamment en cas de situations urgentes. La rigidité des requêtes SPARQL réside dans leur nature statique et strictement définie. Si les questions posées nécessitent une flexibilité imprévue, il devient difficile de réutiliser ou d'adapter les requêtes existantes sans les modifier manuellement. Solution : Agrégation des valeurs et filtrage précoce des données pour améliorer les performances.
- **Formulation complexe des requêtes :** La création et la gestion des requêtes peuvent être difficiles pour les utilisateurs non techniques, limitant l'accessibilité et l'efficacité du système. Solution : Développer une application avec une interface utilisateur intuitive permettant de formuler et de personnaliser les requêtes.
- **Taille de l'ontologie** : Le stockage de toutes les observations pour toutes les valeurs peut rapidement devenir très volumineux. Solution : Supprimer les données anciennes ou non pertinentes pour alléger l'ontologie.
- **Vitesse de génération/collecte des données** : Si trop d'événements sont traités en même temps, un retard peut survenir entre la détection d'une anomalie et l'enregistrement de la situation associée. Solution : Éliminer les données qui ne font plus partie de la fenêtre active, ce qui peut entraîner la perte d'informations passées.

5. Possible applications of this approach in a collaborative learning framework. Points raised in the lectures can be discussed here in more detail.

L'apprentissage collaboratif implique plusieurs entités (experts, modèles, ontologies, etc.) qui apprennent ensemble en partageant leurs connaissances et compétences. Elles collaborent en échangeant des données et des informations, tout en vérifiant et enrichissant mutuellement leurs contributions.

A. **Combinaison de perspectives multiples :**

- Le point de vue des experts est intégré dans l'ontologie (c'est ce qui nous a permis de la modéliser), et les requêtes pour déterminer les situations et contraintes se basent sur leurs connaissances, ce qui a été utilisé dans notre implémentation. *Expert (domain) knowledge integration*
- D'un autre côté, notre scénario pourrait aider à l'enrichissement des savoirs des experts. On utilise le *stream reasoning* pour enrichir les connaissances des experts (cf. les approches B, C, D page suivante).
- Notre scénario, qui représente le point de vue expert, peut être combiné avec ceux de plusieurs modèles pour une évaluation plus complète. Ce processus permet de combiner les prédictions des différents modèles et de guider la prise de décision finale, en apportant des perspectives multiples pour une gestion plus efficace des anomalies. *Help in final decision making combining predictions from multiple models.*

B. **Amélioration des modèles via l'ontologie :** L'ontologie, enrichie par des connaissances expertes, peut être utilisée pour fournir des explications ou des insights supplémentaires aux experts sur leurs données en temps réel. *Instead of sharing raw data, clients can share model updates or predictions that are enriched with domain knowledge.*

C. **Découverte d'anomalies inconnues :** Grâce à l'apprentissage collaboratif, les modèles peuvent identifier des anomalies qui échappent aux experts (et donc à l'ontologie), en ajustant les contraintes (par exemple les seuils) et en découvrant de nouvelles situations.

D. **Génération de situations pour l'apprentissage des modèles :** Utiliser notre implémentation comme générateur pour créer de nouvelles situations labellisées que les modèles peuvent apprendre à reconnaître. Ce processus pourrait être similaire à un GAN, où le *stream reasoning* serait utilisé pour générer des données et le modèle devrait apprendre à les classer (améliorer la détection d'anomalie).

E. **Validation des sorties des modèles prédictifs :** En utilisant, les séries temporelles entrées dans les modèles, labellisées en sortie des modèles, valider (vérifier la cohérence) avec notre implémentation les prédictions des modèles.

F. **Changement de représentation des données :** Utiliser notre implémentation pour transformer (et réduire les données) des séries temporelles complexes en données catégorielles (par exemple, des colonnes représentant des situations spécifiques), facilitant ensuite l'apprentissage des modèles. **Par exemple,** l'utilisation de SAX (Symbolic Aggregate approXimation) : Convertir des séries temporelles en représentations symboliques, puis utiliser ces représentations simplifiées pour améliorer l'apprentissage des modèles, en réduisant la complexité des données à traiter.

G. **Utilisation des graphes de connaissances pour guider l'apprentissage.** Par exemple, ils pourraient permettre de relier différentes séries temporelles et de montrer comment elles "interagissent", ce qui faciliterait la coordination entre les modèles. Cela pourrait permettre d'améliorer la précision des prédictions en intégrant les connaissances partagées, et au lieu de simplement échanger des données brutes, on peut échanger des informations plus pertinentes et adaptées au contexte. *Use of knowledge graphs to represent relations between data, entities, or features may help guide the federated learning process.*

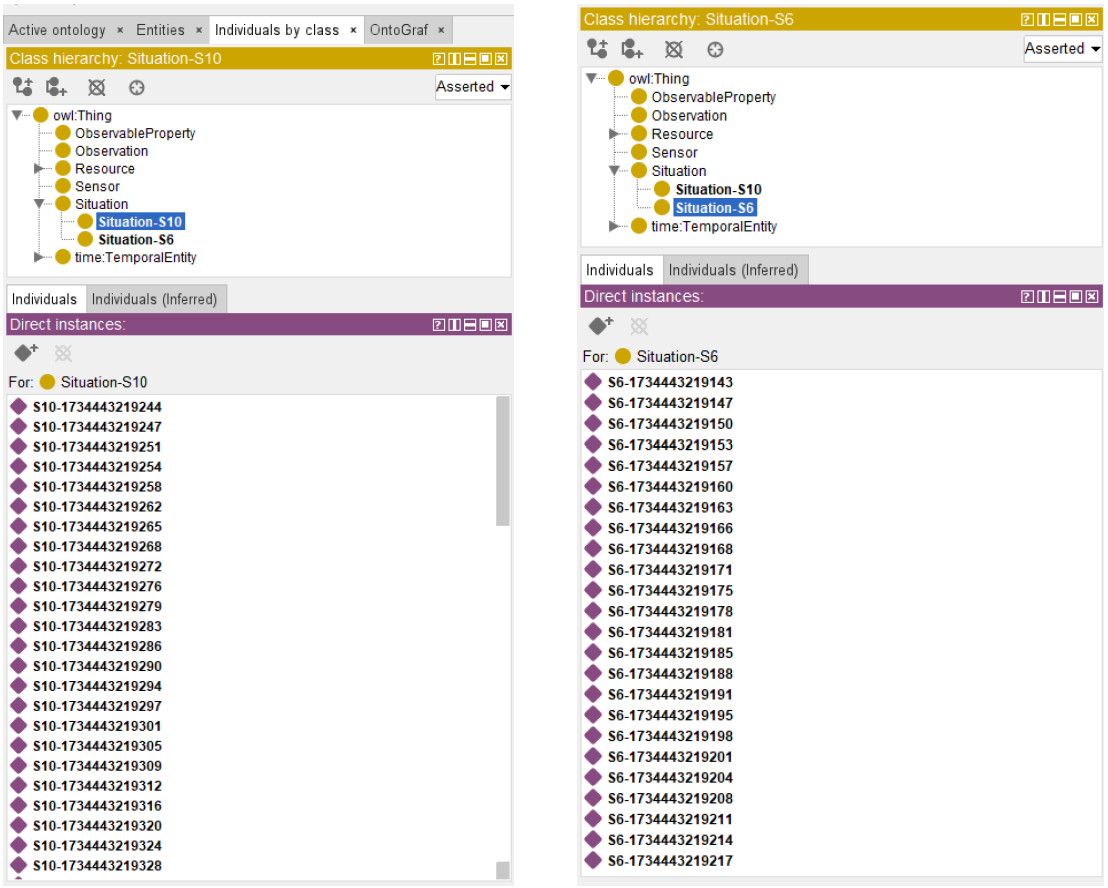
Annexe :

```
REGISTER QUERY S6detection AS
  PREFIX : <http://semanticweb.org/Ontology-TP#>
  PREFIX sosa: <http://www.w3.org/ns/sosa/>

  SELECT ?m ?pl
  FROM STREAM <Stream_S_ConvWaterTemp> [RANGE 15s STEP 5s]
  FROM STREAM <Stream_S_TransGridTemp> [RANGE 15s STEP 5s]
  FROM STREAM <Stream_S_GeneratorTemp> [RANGE 15s STEP 5s]
  FROM <http://streamreasoning.org/Ontology-TP>

  WHERE {
    ?m         :isPartOf        ?pl .
    ?m         sosa:hosts       sosa:S_ConvWaterTemp .
    ?m         sosa:hosts       sosa:S_TransGridTemp .
    ?m         sosa:hosts       sosa:S_GeneratorTemp .
    :S_ConvWaterTemp :madeObservation ?o1 .
    :S_TransGridTemp :madeObservation ?o2 .
    :S_GeneratorTemp :madeObservation ?o3 .
    ?o1        :hasSimpleResult ?v1 .
    ?o2        :hasSimpleResult ?v2 .
    ?o3        :hasSimpleResult ?v3 .

    FILTER (
      ?v1 > 60 &&
      ?v2 < 35 &&
      ?v3 > 45
    )
  };
```

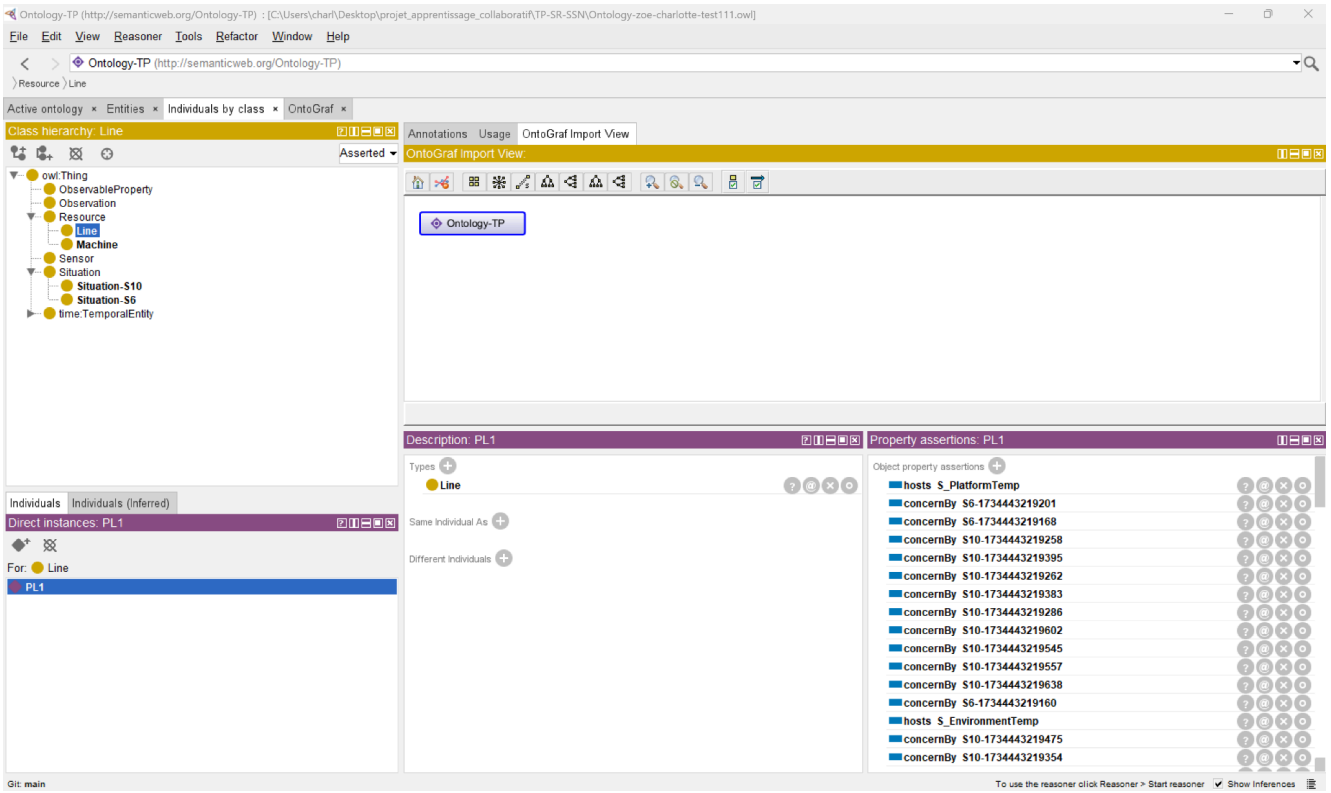
Annexe 1 : Requête C-SPARQL pour détection de la situation 6

```
App.java
ConsoleFormatter.java
Ontology-zoe-charlotte.owl  # ontologie avant de lancer stream reasoning
Ontology-zoe-charlotte-test-output.owl  # ontologie après, on y voit des
situations insérées
App_demo.java  # le fichier dans demo/.../App.java du TP qui a servi à générer
l'ontologie initiale
```

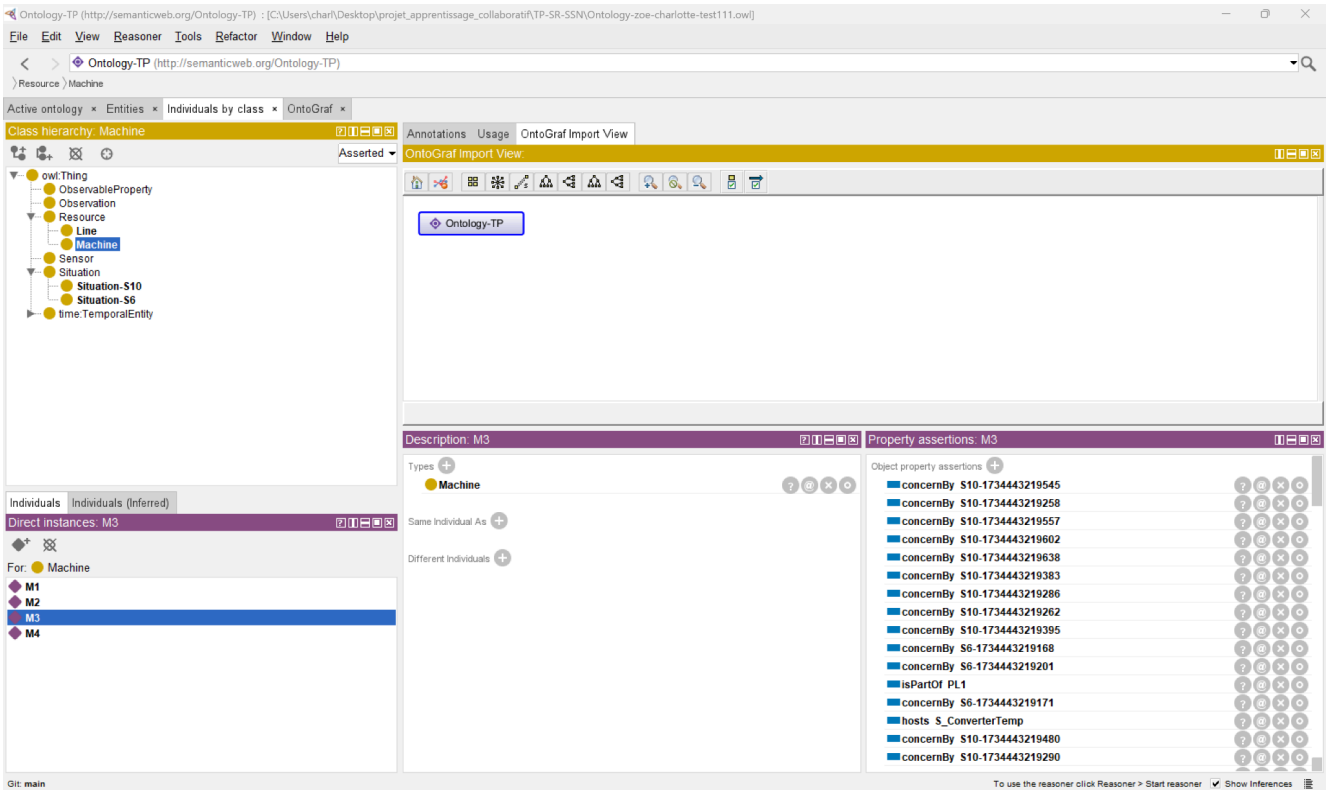
Annexe 2 : Fichiers et explications



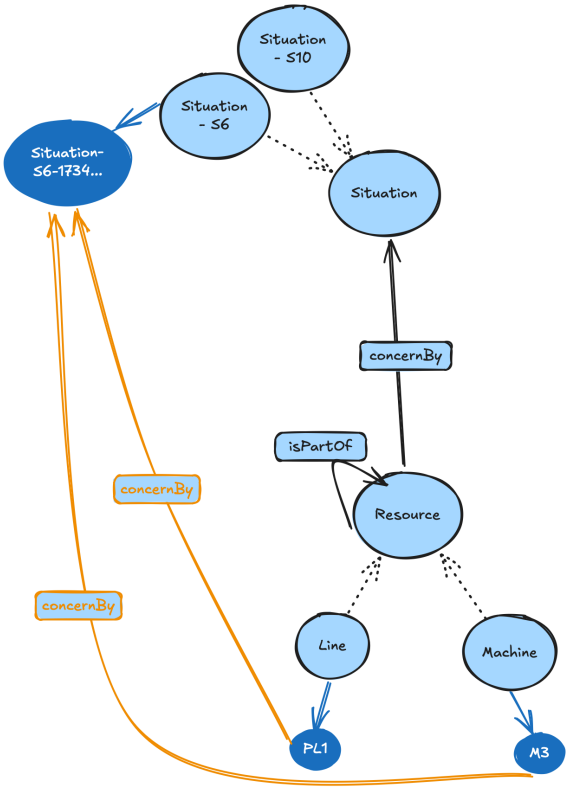
Annexe 3 : Situations détectées, insérées dans l'ontologie



Annexe 4 : Liens (*concernBy*) entre situations détectées et ligne de production dans l'ontologie



Annexe 5 : Liens (*concernBy*) entre situations détectées et machine dans l'ontologie



Annexe 6 : Schéma de l'ontologie d'un ajout de situation détectée sur la machine M3 et la ligne de production PL1