Formalisation du concept d'affordance dans l'ontologie Thing Description

Victor Charpenay

Formalisation du concept d'affordance dans l'ontologie Thing Description

V. Charpenay
Laboratoire d'informatique, de modélisation et d'optimisation des systèmes (LIMOS)

victor.charpenay@emse.fr

Résumé

Cet article présent l'ontologie Thing Description (TD), une ontologie pour décrire objets connectés et systèmes cyber-physiques sur le web. L'ontologie TD formalise le concept d'« affordance » comme la relation qui existe entre une requête HTTP envoyée par un agent, la réponse qui sera donnée par le serveur et les effets de cet échange agent/serveur sur le monde physique.
Les axiomes de l'ontologie TD, qui font intervenir des termes des ontologies Semantic Sensor Networks et Smart Applications Reference, sont évalués vis-à-vis de la « commandabilité » de l'objet ou du système décrit. Une formalisation logique de cette notion issue de la théorie du contrôle est proposée dans l'article.

Mots-clés

Web des objets, Thing Description, commandabilité, système cyber-physique, planification automatique.

Abstract

This paper presents the Thing Description (TD) ontology, an ontology to describe connected devices and cyber-physical systems on the Web. The TD ontology formalizes the concept of 'affordance' as the relationship between an HTTP request sent by an agent, the response provided by the server and the effects of the agent/server communication in the physical world.
The axioms of the TD ontology, involving terms from the Semantic Sensor Networks and Smart Applications Reference ontologies, are evaluated against the 'controllability' of the device or system being described. The paper introduces a logical formalization of this notion inherited from control theory.

Keywords

Web of Things, Thing Description, Controllability, Cyber-Physical System, AI Planning.

1 Introduction

Le web des objets a pour principe fondamental d'exposer des objets connectés à travers une interface web, dite « REST » (pour *Representational State Transfer*) [11]. Cette approche apporte principalement des avantages en termes d'ingénierie logicielle mais à l'heure actuelle, il n'existe pas de formalisme théorique établi pour REST. Notamment, l'approche REST invoque la notion d'« affordance », qui n'a pourtant pas de fondement théorique dans le contexte du web. Ce néologisme, qui émane du champ de la psychologie et du design [10], désigne la capacité d'un objet à suggérer son usage. Sur le web, les objets en questions sont des documents, qu'un client manipule à travers des hyperliens (pour y accéder) et des formulaires web (pour modifier leur état). Hyperliens et formulaires web seraient donc les affordances des documents qui suggèrent au client comment manipuler un document en particulier [5].
Cette notion d'affordance est prépondérante pour permettre à des clients de contrôler les objets exposés sur le web de manière autonome. Les affordances offertes par hyperliens et formulaires web doivent être non seulement comprises des humains mais aussi d'agents logiciels, qui agissent en navigant d'une ressource à l'autre et en remplissant des formulaires.
Dans cet article, nous proposons un cadre formel pour les affordances exposées sur le web par des objets connectés. Ce cadre formel se présente comme une ontologie, dont les termes sont déjà utilisés dans le standard *Thing Description* (TD) du *World Wide Web Consortium* (W3C) [8]. L'objectif principal de l'ontologie TD est de permettre à des agents autonomes d'appliquer les techniques d'intelligence artificielle (IA) classiques, telles que du raisonnement sur des situations ou de la planification, à des descriptions d'objets connectés et, plus largement, à des systèmes cyber-physiques.
Comme tout ontologie du web, l'ontologie TD cherche à réutiliser les ontologies existantes, notamment celles standardisées par le W3C ou d'autres instituts de normalisation. L'ontologie TD est donc définie par rapport aux ontologies *Semantic Sensor Network* (SSN), *OWL Time*, *HTTP in RDF* et l'ontologie de provenance PROV-O, toutes du W3C, ainsi que *Smart Applications Reference* (SAREF), un standard de l'institut européen des normes de télécommunication. Voir table 1 pour les *namespaces* associés à chacune de ces ontologies.
Cet article prolonge un article de 2020 qui présentait l'ontologie TD vis-à-vis de questions de compétences précises, posées par le groupe de travail W3C autour du web des objets [4]. Alors que l'article de 2020 basait son évaluation (entre autres) sur une tâche unique de « sélection d'affordance », le travail suivant présente une généralisation des tâches possibles que des agents autonomes peuvent effec-

TABLE 1 – Ontologies référencées dans l'article

Ontologie	*Namespace*
SSN	`http://www.w3.org/ns/ssn/`
OWL Time	`http://www.w3.org/2006/time#`
HTTP in RDF	`ttp://www.w3.org/2011/http#`
PROV-O	`http://www.w3.org/ns/prov#`
SAREF	`https://saref.etsi.org/core/`
TD	`https://www.w3.org/2019/wot/td#`

tuer sur des systèmes cyber-physiques. Cette généralisation est basée sur la « commandabilité » des systèmes, une notion issue de la théorie du contrôle.
La notion de commandabilité, ainsi que d'autres préliminaires sont définis dans la suite de l'article (partie 2), qui donne ensuite les axiomes de l'ontologie, après généralisation vis-à-vis l'article de 2020 (partie 3) et un exemple complet d'instantiation de l'ontologie (partie 4). L'article se termine sur une synthèse de l'approche (partie 5).

2 Préliminaires

2.1 Actions et temporalité

Les techniques d'IA classiques sont toutes basées sur une représentation formelle du temps et/ou des actions effectuées par des agents autonomes. De nombreuses théories du temps ont été proposées, comme la logique temporelle linéaire (LTL) [9], beaucoup utilisée pour de la vérification formelle, ou l'algèbre d'intervalles d'Allen [2]. Ces deux approches utilisent des modèles différents ; la première représente le temps comme un ensemble discret de points, la seconde comme un ensemble d'intervalles. Il existe des équivalentes, cependant, selon les extensions syntaxiques considérées [6].
Dans le contexte du web sémantique, l'outil logique de référence est OWL, le langage des ontologies web. OWL est en revanche très rarement utilisé pour faire du raisonnement temporel ou de la planification. La raison, bien qu'il n'en existe pas de preuve formelle à notre connaissance, tiendrait au fait que le formalisme sous-jacent à OWL (la logique de description $\mathcal{SROIQ}$) ne permet d'encoder ni les logiques temporelles par points comme LTL, ni l'algèbre d'Allen [1].
Les travaux de recherches se rapprochant le plus de OWL pour représenter temporalité et actions ont été effectués par Artale et Franconi [3] et sont basés sur $\mathcal{HS}$ [7], une logique plus expressive mais indécidable (pour le problème de satisfaction). Artale et Franconi reprennent l'algèbre d'Allen comme modèle temporel de base. Les termes de cette algèbre sont définis dans l'ontologie du W3C OWL Time [2].
Dans la suite de l'article, nous utiliserons OWL Time comme modèle temporel de base pour les axiomes de l'ontologie TD. Par souci de lisibilité, les axiomes seront exprimés dans un forme plus classique, en tant que formules

1. des opérateurs non-standards de logique de description, incompatibles avec $\mathcal{SROIQ}$ d'un point de de la décidabilité du raisonnement, permettent cependant de le faire : un opérateur de point fixe pour LTL, et la disjonction de roles pour l'algèbre d'Allen.

2. mais la sémantique de l'algèbre, elle, n'est pas complètement axiomatisée dans l'ontologie.

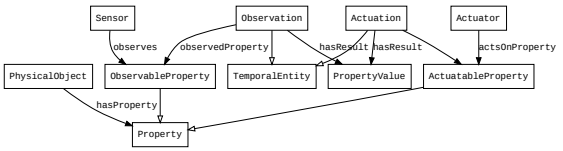

FIGURE 1 – Vocabulaire et axiomes RDFS (hiérarchies de classes, domaines/images de propriétés) d'une ontologie de base pour le web des objets

de logique de premier ordre dans la syntaxe *Common Language Interchange Format* (CLIF) [1]. L'équivalence avec la logique proposée par Artale et Franconi ne sera pas traité ici.
L'algèbre d'Allen (et donc OWL Time) définit treize relations possibles entre intervalles de temps. Si `t1` et `t2` sont des intervalles de temps, alors on peut par exemple définir `(before t1 t2)`, `(meets t1 t2)`, `(overlaps t1 t2)` ou `(equals t1 t2)`. La relation `equals`, ici, n'est pas une égalité stricte. `t1` et `t2` coïncident dans le temps mais ces deux intervalles peuvent représenter des événements distincts.
Les observations faites par des capteurs ou les activations effectuées par des actionneurs sont typiquement les intervalles sur lesquels raisonner dans le web des objets. Ces termes sont présentés plus en détail dans la partie suivante.

2.2 *Semantic Sensor Networks*

L'ontologie TD se base sur un ensemble de termes préexistants, pour la plupart tirés du standard W3C *Semantic Sensor Network* (SSN). Les termes de cette ontologie de base sont synthétisée en un diagramme (figure 1), dont le langage graphique utilise les notions courantes du web sémantique.
Les classes (prédicats unaires) apparaissent comme des rectangles et les relations [3] (prédicats binaires) apparaissent à travers des connexions entre classes. Une connexion non-labelisée implique une hiérarchie de classe. C'est par exemple le cas entre `Observation` et `TemporalEntity` (la classe de tous les intervalles de temps). Une connexion labélisée implique, elle, des axiomes de domaine(s) et d'image(s). La relation `observedProperty` est par exemple définie entre les classes `Observation` et `ObservableProperty`.
Ces types d'axiomes, à la base de RDF Schema (RDFS), permettent principalement de définir des structures de données en graphe. À travers cet article, nous ajoutons d'autres axiomes aux termes de SSN dans le cadre du web des objets. Ces axiomes sont exprimés en tant que formules en logique de premier ordre, comme discuté précédemment. Une distinction sera faite entre les formules ouvertes (à voir comme des motifs syntaxiques récurrents), contenant des variables libres non-quantifiées, et les formules fermées (les

3. plus généralement appelées « propriétés » dans la terminologie RDF. Le terme est cependant ambigü ici car il désigne aussi les propriétés d'objets physiques.

axiomes eux-mêmes).
Par exemple, l'ontologie SSN définit les classes `Observation` et `Actuation`, qui associent une propriété physique à une valeur observée ou fixée par un agent. Les formules suivantes matérialisent ces associations, ce sont des formules ouvertes :

(OBS)

```
(and
  (Observation o)
  (observedProperty o p)
  (hasResult o v))
```

(ACT)

```
(and
  (Actuation a)
  (actsOnProperty a p)
  (hasResult a v))
```

Les variables libres apparaissant dans ces formules ouvertes sont ensuite quantifiées dans les deux axiomes suivants (par souci de concision, les classes `TemporalEntity`, `Property` et `PropertyValue`) ont été raccourci en `T`, `P` et `V`) :

```
(forall ((T o) (P p) (V v) (V v'))
  (if OBS_{o,p,v}
    (not (exists (T o')
      (and
        OBS_{o',p,v'}
        (finishes o' o) (not (= v v')))))))

(forall ((T a) (T o) (P p) (V v) (V v'))
  (if
    (and
      ACT_{a,p,v}
      OBS_{o,p,v'}
      (or (before a o) (meets a o))
      (not (exists (T a')
        (and
          ACT_{a',p,v''}
          (or (before a a') (meets a a'))
          (or (before a' o) (meets a' o))
          (not (= v v'')))))))
    (= v v')))
```

Les axiomes ci-dessus expriment respectivement que deux observations faites pour une même propriété au même instant doivent avoir le même résultat et que toute observation doit avoir le même résultat que la dernière activation la précédent.
L'axiome suivant donne la contrainte qu'une propriété doit être associée à une valeur en tout instant :

```
(forall (T o) (P p)
  (if OBS_{o,p,v}
    (exists (T o') (V v')
      (and (meets o o') OBS_{o',p,v'}))))
```

Ensemble, ces axiomes permettent de réduire les systèmes décrits avec SSN à un ensemble de lignes de temps (une pour chaque propriété physique du système) sur lesquelles chaque point, correspondant à un instant donné, est la valeur de propriété mesurable par un capteur à cet instant. Ces axiomes ne font pas partie des ontologies dont nous empruntons le vocabulaire, ils serviront cependant à valider formellement l'intérêt de l'ontologie TD pour diverses techniques d'IA.
Dans l'article original de 2020 sur l'ontologie TD, plus de la moitié des objets décrit dans le jeu de données utilisée pour l'évaluation [4] servait à contrôler un système d'éclairage. L'exemple suivant en est inspiré.

Exemple 1. *Les formules suivantes décrivent un interrupteur on/off et un capteur d'illuminance, à trois niveaux de luminosité (obscur, moyen, clair).*

```
(and
  (PhysicalObject switch)
  (hasProperty switch state)
  (ObservableProperty state)
  (ActuatableProperty state)
  (forall (T t)
    (if (or OBS_{t,state,v} ACT_{t,state,v})
      (or
        (= v on)
        (= v off)))))

(and
  (Sensor lightSensor)
  (hasProperty lightSensor level)
  (ObservableProperty level)
  (not (ActuatableProperty level))
  (forall (T t)
    (if (or OBS_{t,level,v} ACT_{t,level,v})
      (or
        (= v dark)
        (= v average)
        (= v bright)))))
```

À partir de l'ontologie de base présentée ici, il serait possible de définir des lois physiques. Un exemple en sera donné par la suite. La caractérisation complète des lois qui régissent observations et activations vont cependant au-delà du périmètre de cet article. Le fondement de l'article est ailleurs, dans la commandabilité des systèmes cyber-physiques.

2.3 Commandabilité

La notion de commandabilité est une notion importante de la théorie du contrôle [12] car elle caractérise simplement en termes matriciels l'efficacité d'une procédure de contrôle dans des systèmes linéaires. Nous la redéfinissons ici dans un cadre logique comme la satisfiabilité d'une formule de précédence entre un état initial donné et un état final souhaité (état cible).

Définition 1. *Soit S une formule spécifiant le comportement d'un système à base d'objets physiques à partir d'un intervalle de temps `now`. Soit ϕ_t une formule temporelle avec la variable libre `t` telle que `(T t)`. ϕ_t est un « état cible » pour lequel est définie la formule suivante :*

4. https://w3c.github.io/wot-thing-description/testing/report.html

(K)

```
(forall (T t')
  (if (before now t')
    (exists (T t)
      (and ϕ_t (before t' t)))))
```

Le système spécifié par S est contrôlable si et seulement si la formule $S \wedge K$ est satisfiable, c'est-à-dire qu'il existe un modèle de premier ordre M tel que :

$$M \models S \wedge K$$

Cette caractérisation logique de la notion de commandabilité peut intuitivement se formuler ainsi : un système est contrôlable s'il est possible d'atteindre l'état cible depuis l'état initial. L'état cible n'est cependant pas atteint en toute circonstances (les propriétés activables ont en effet plusieurs activations possibles). C'est le rôle d'agents autonomes d'effectuer les actions nécessaires pour atteindre cet état.
Par ailleurs, la formule K nécessite que l'intervalle `t` soit atteignable depuis n'importe quel état depuis l'état initial (cf `forall (T t')`). Cette condition permet de considérer le cas ou des changements spontanés s'opèrent sur les objets une fois l'état cible atteint, du fait de lois physiques. Cette notion généralisée de commandabilité est aussi appelée « stabilisabilité » dans la littérature sur la théorie du contrôle.
Les états cibles qui nous intéressent dans le cadre du web des objets sont ceux définis avec l'ontologie de base présentés dans la partie précédente, par exemple à travers l'observation d'une valeur particulière. En reprenant l'exemple 1, on peut par exemple définir comme état cible une observation où l'éclairage est au plus haut : $\text{OBS}_{t,\texttt{level},\texttt{bright}}$.

3 Axiomes

La finalité de l'ontologie TD est de décrire l'interface web permettant à un agent logiciel autonome d'interagir avec des objets physiques, c'est-à-dire d'en observer les propriétés et les contrôler. Dans le web des objets, l'action des agents sur les objets prend la forme d'opérations effectuées sur des ressources web à travers des formulaires.
Dans le formalisme présenté dans cet article, tous les atomes apparaissant dans les formules (par exemple, `switch`, `state` ou `off`) sont considérés comme des ressources web, dont une représentation concrète peut être déréférencée par les agents. L'ontologie TD définit des axiomes permettant aux agents de connaître les implications sur un objet physique d'une opération web.
Le diagramme donnant le vocabulaire et les axiomes RDFS de l'ontologie TD (figure 3) inclut des termes importés d'autres ontologies (*HTTP in RDF* et PROV-O, pour la relation `wasGeneratedBy`). Il est à noter que la classe `Thing`, elle, appartient au *namespace* de l'ontologie TD. Elle est distincte de son homonyme définie par OWL, la classe mère du langage.
Les termes importés permettent de représenter par des formules les messages échangés entre agents et serveurs web. Ces messages sont à la base des opérations web.

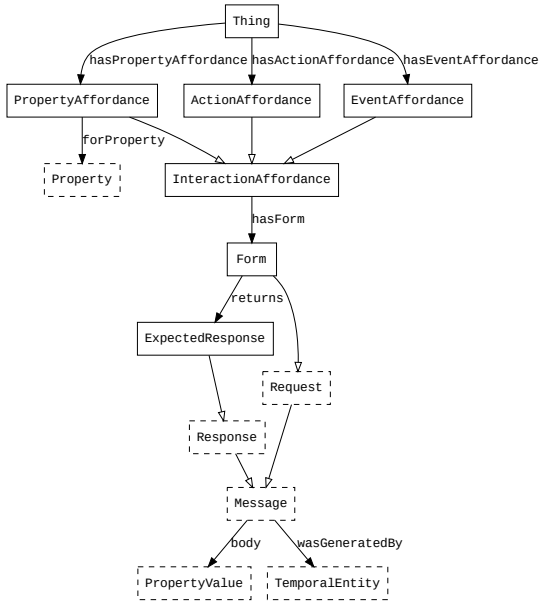

FIGURE 2 – Vocabulaire et axiomes RDFS de l'ontologie TD (vocabulaire extérieur en pointillé)

3.1 Opérations et affordances

Une opération est l'intervalle inscrit entre l'envoi d'une requête et celui d'une réponse. Dans le cadre du web des objets, les opérations sont rarement matérialisées, l'ontologie TD n'inclut donc pas de vocabulaire spécifique pour les opérations. On peut cependant caractériser une opération en n'utilisant que le vocabulaire d'*OWL Time*, comme ci-dessous.

(OP)

```
(and
  (Request req) (Response resp)
  (meets t1 op) (isMetBy t2 op)
  (wasGeneratedBy req t1)
  (wasGeneratedBy resp t2)
  (not (exists (T t2')
    (and
      (before t2' t2)
      (wasGeneratedBy resp t2')))))
```

Une représentation schématique des intervalles `t1`, `op` et `t2` est donnée en figure 3. Dans la formule définissant `op`, la non-existence d'un intervalle `t2'` garantit l'unicité de `op` : seule la première réponse succédant à la requête est prise en compte.
À partir de la notion d'opération, on peut maintenant définir formellement ce qu'est une affordance.

(AF)

```
(and
  (InteractionAffordance alpha)
  (hasForm alpha req)
  (returns req resp))
```

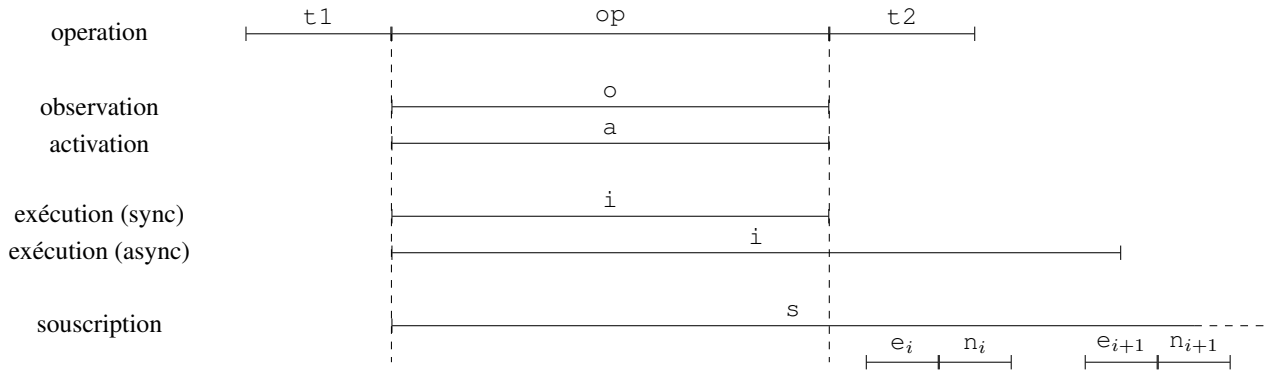

FIGURE 3 – Representation schématique des axiomes liés aux affordances sur des opérations REST

Dans la structure ci-dessus, l'affordance est associée à une requête (un formulaire web, *form* en anglais) et une réponse particulière. La notion de formulaire web est ici à prendre au sens large. Il ne s'agit pas que d'un simple formulaire HTML mais d'un patron de requête qu'un agent web peut envoyer à un serveur.
Dans sa version la plus simple, une `InteractionAffordance` spécifie quelle réponse un agent obtiendra en soumettant un formulaire.

Définition 2. *Une affordance est la relation qui existe entre une requête REST potentielle (un formulaire web) et la réponse attendue à cette requête :*

```
(forall (alpha req resp)
  (if AF_{alpha,req,resp}
    (forall (T t1)
      (if (wasGeneratedBy req t1)
        (exists ((T op) (T t2))
          OP_{op,t1,t2,req,resp})))))
```

Cependant, cette définition n'est pas suffisante pour permettre à des agents autonomes d'agir de manière informée. Ce type de connaissance peut déjà être formalisé à travers des descriptions de services web (qui décrivent les structures de données en entrée et en sortie d'un service). OpenAPI [5] est par exemple une spécification normalisée qui uniformise la description d'interfaces REST.
La notion d'affordance provient de recherches en psychologie et en design. Dans sa définition originelle, proposée par James Gibsons, une affordance est rattachée à un objet, elle est fournie par lui à un agent comme action potentielle. Sur le web, on a comparé le concept d'hyperlien à une affordance, en particulier dans son rendu visuel (curseur engageant à cliquer). Sur le web des objets, les objets physiques fournissent des affordances aux agents à travers la classe d'`InteractionAffordance`.
Les sous-classes d'`InteractionAffordance` permettent d'axiomatiser plus en détail l'effet d'une opération sur le monde physique et, ainsi, d'inférer des plans d'actions sur la base de la commandabilité des objets décrits.

5. https://www.openapis.org/

Il est à noter que la classe `Thing`, qui regroupe toutes les entités physiques fournissant des affordances, est distincte de celle de `PhysicalObject`. Dans l'architecture du web des objets, l'objet qui fournit l'affordance n'est en effet pas nécessairement l'objet d'une action. Le type d'affordance présenté dans la partie suivante permet de faire la distinction.

3.2 Observations et activation simples

La principale sous-classe d'`InteractionAffordance` définit des affordances associées à une propriété physique.

(PAF)

```
(and
  (PropertyAffordance alpha)
  (forProperty alpha p))
```

Définition 3. *Une affordance de propriété est l'affordance de lire ou écrire la valeur d'une propriété physique :*

```
(forall (alpha req resp p)
  (iff
    (and
      AF_{alpha,req,resp} PAF_{alpha,p}
      (ObservableProperty p))
    (forall ((T op) (T t1) (T t2))
      (if OP_{op,t1,t2,req,resp}
        (exists (T o)
          (and
            OBS_{o,p,v}
            (equals op o)
            (body resp v)))))))

(forall (alpha req resp p)
  (iff
    (and
      AF_{alpha,req,resp} PAF_{alpha,p}
      (ActuatableProperty p))
    (forall ((T op) (T t1) (T t2) (V v))
      (if (and OP_{op,t1,t2,req,resp} (body req v))
        (exists (T a)
          (and
            ACT_{a,p,v}
            (equals op a)))))))
```

Comme mentionné précédemment, on distingue `Thing` et `PhysicalObject` et, de la même manière, on distingue `PropertyAffordance` et `Property`. Par exemple, si l'appareil `lightSensor` fournit une affordance pour la propriété `level`, le niveau de luminosité observé n'est pas la propriété du capteur mais celle de la pièce dans laquelle il se trouve. Un autre capteur situé dans la même pièce pourrait fournir une affordance distincte pour la même propriété. Par ailleurs, la pièce elle-même pourrait fournir une affordance sur sa propre luminosité, masquant ainsi l'infrastructure technique d'acquisition de données.
Les questions de compétences de notre article de 2020 élicitaient le besoin formulé par le W3C d'identifier différentes combinaisons possibles entre ces classes. Outre le cas où deux affordances s'appliquent à la même propriété, il est aussi possible qu'une même affordance s'applique à plusieurs propriété (par exemple lorsqu'une station météo fournit une mesure de température et d'humidité dans le même message).

3.3 Invocations et souscriptions

Les affordances pour observer ou agir sur les propriétés physiques sont les plus utilisées en pratique (84% des 371 affordances incluses dans notre jeu de données de 2020). Cependant, elles ne suffisent pas complètement à permettre à tout système d'être contrôlable à travers le web. D'autres formes plus complexes d'affordances sont nécessaires : `ActionAffordance` et `EventAffordance`.
Intuitivement, lorsque une propriété change de manière continue et rapide, le délai introduit par le fait d'envoyer des requêtes REST (discrètes) pour l'observer ne permet pas toujours à des agents de réagir à temps. Une requête peut cependant exiger de l'objet d'effectuer lui-même une action (s'il s'agit d'un système cyber-physique) ou de notifier l'agent lors d'événements particuliers. C'est l'intérêt des deux sous-classes d'affordances présentées dans cette partie.
Une affordance d'action est associée à une action, qui accepte des paramètres en entrée et qui retourne une valeur en sortie.

(AAF)

```
(and
  (ActionAffordance alpha)
  (hasInputSchema alpha schi)
  (hasOutputSchema alpha scho))
```

Définition 4. *Une affordance d'action est l'affordance d'invoquer l'exécution d'une action :*

```
(forall (alpha req resp schi scho)
  (iff
    (and AF_{alpha,req,resp} AAF_{alpha,schi,scho})
    (forall ((T op) (T t1) (T t2))
      (if (and OP_{op,t1,t2,req,resp} (body req schi))
        (exists (T i)
          (and
            (forInvocation alpha i)
            (or
              (and
                (equals op i)
                (body resp scho))
              (and
                (starts op i)
                (body resp i)))))))))
```

Une `ActionAffordance` peut modéliser l'un des deux motifs récurrents dans les interfaces REST, selon le modèle d'éxecution de l'action invoquée : synchrone ou asynchrone. Dans le premier cas, l'invocation dure le temps de l'opération. Elle commence à la réception de la requête et se termine à l'envoi de la réponse. Dans le second cas, l'invocation peut se prolonger au-delà de l'opération. L'agent peut différencier ces deux motifs grâce à la nature de la réponse obtenue (une valeur de sortie `scho` ou l'invocation `i` elle-même, en tant qu'entité temporelle). En pratique, lorsque l'invocation est asynchrone, déréfencer l'entité temporelle retournée permet de connaître le statut de l'invocation (« en cours », « terminée », « annulée », ou autres).
Il est important ici de distinguer deux types d'actions en jeu dans des affordances. Les actions invoquées par un agent sont effectuées par l'objet ou système sous contrôle. Cependant, les invocations elles-mêmes sont des actions (de délégation) qui sont, elles, effectuées par l'agent. Dans un problème de planification, par exemple, les actions à planifier sont celles de l'agent, donc toutes les opérations qui lui sont offertes à travers des affordances, et pas uniquement les actions invocables à travers une affordance d'action.
Une affordance d'événement joue un rôle analogue à l'affordance d'action. Si l'affordance d'action permet de paramétrer des activations, l'affordance d'événement permet de paramétrer les observation faites par les agents.

(EAF)

```
(and
  (EventAffordance alpha)
  (hasSubscriptionSchema alpha schs)
  (hasNotificationSchema alpha schn))
```

Définition 5. *Une affordance d'événement est l'affordance de souscrire à un événement afin de recevoir une notification à chaque occurrence de l'événement :*

```
(forall (alpha req resp schs schn)
  (iff
    (and AF_{alpha,req,resp} AAF_{alpha,schs,schn})
    (forall ((T op) (T t1) (T t2))
      (if (and OP_{op,t1,t2,req,resp} (body req schs))
        (exists (T s)
          (and
            (forSubscription alpha s)
            (starts op s)
            (body resp s)))))))
```

On peut noter que l'axiome ci-dessus n'utilise pas `schn` (le schéma de notification). En effet, pour pouvoir axiomatiser qu'une notification est envoyée à chaque événement, il est d'abord nécessaire de caractériser l'événement, en

termes d'observations et/ou d'activations. Même sans caractériser l'action ou de l'événement en jeu, pourtant, l'intérêt d'axiomatiser des affordances d'action ou d'événement est d'établir l'existence d'entités temporelles (l'invocation `i` et la souscription `s`, respectivement) qui peuvent ensuite servir à formuler des contraintes plus spécifiques sur des sous-classes d'affordance. Afin de relier une affordance à `i` ou `s`, on utilise les relations `forInvocation` et `forSubscription`. Nous en donnerons un exemple dans la partie suivante. Les axiomes complets pour ces relations ne sont cependant pas trancrites ici.
Dans le cas d'une affordance d'événement, toute caractérisation de l'événement devrait utiliser la formule suivante pour décrire une notification (avec la variable libre `e`) :

(N)

```
(exists ((T n) m)
  (and
    (meets e n)
    (Message m)
    (wasGeneratedBy m n)
    (body m schn)))
```

L'ontologie TD telle que publiée par le W3C inclut aussi un schéma pour annuler une souscription. Cette partie n'est pas détaillée ici.

3.4 Boucle de commande

Le théorème suivant garantit que l'ajout des classes `ActionAffordance` et `EventAffordance` à `PropertyAffordance` suffit à décrire tout type de système sur le web (à la condition que le système décrit est commandable, pour un état cible défini à travers l'ontologie de base du web des objets).

Théorème 1. *On suppose que chaque activation est précédée d'une observation (boucle de commande) et que, par ailleurs, il ne peut y avoir qu'une seule opération à un instant donné (éxecution linéaire).*
Dans ces conditions, les trois types d'affordances de propriété, d'action et d'événement sont suffisants pour représenter tout système contrôlable sur le web.

Démonstration. (esquisse) Considérons le modèle M qui, par hypothèse, satisfait $S \wedge K$.
On suppose l'existence d'une activation `a1`, nécessaire à la commandabilité. On construit M' tel qu'une nouvelle opération `op1` englobe l'activation (`(contains op1 a1)`). On définit ensuite une affordance `alpha`, telle que `(PropertyAffordance alpha)`, qui garantisse l'existance d'`op1`.
S'il existe une autre activavtion `a2` telle que `(meets a1 a2)`, on définit `alpha` telle que `(ActionAffordance alpha)`, de telle sorte que l'action invoquée englobe `a1` et `a2`.
On fait de même avec les observations qui précèdent `a1` et `a2`. Si une observation `o1` n'est pas collée à `a1`, on définit une affordance de propriété. Sinon (`(meets o1 a1)`), on définit une affordance d'événement. □

Les trois types d'affordances ne sont en revanche pas nécessaires à garantir la commandabilité du système décrit. Certains systèmes définissent uniquement des affordances d'action pour modifier la valeur d'une propriété activable. La possibilité d'écrire une propriété à travers une `PropertyAffordance` est donc redondant avec la définition d'`ActionAffordance`. C'est pour permettre de modéliser des systèmes particuliers, comme les systèmes de gestion de bâtiments BACnet, que ce choix de la redondance a été fait par le W3C. Les objets BACnet ne définissent que des affordances de propriété ; une modélisation par affordances d'actions serait fastidieux.
Si les axiomes liés aux affordances servent de base à un raisonnement sur la temporalité des actions et des événements, décrire des objets et systèmes physiques avec l'ontologie TD se fait avec les seules formules PAF, AAF, et EAF.

Exemple 2. *Les formules suivantes donne la description TD de l'interrupteur et du capteur d'illuminance de l'exemple 1.*

```
(and
  (Thing switch)
  (hasPropertyAffordance switch paf1)
  (forProperty paf1 state)
  (hasActionAffordance switch aaf))


(and
  (Thing lightSensor)
  (hasPropertyAffordance switch paf2)
  (forProperty paf2 level)
  (hasEventAffordance lightSensor eaf))
```

4 Instantiation

L'objectif de l'ontologie TD est de simplifier la description d'objets ou systèmes physiques tout en garantissant l'application de techniques d'IA pour des agents autonomes sur le web. Les techniques particulières à appliquer ne rentrent cependant pas dans le cadre de l'article, uniquement axé sur la représentation de la connaissance liée aux objets connectés.
Comme mentionné dans la présentation de `ActionAffordance` et `EventAffordance`, une axiomatisation complète sans restriction sur les actions ou événements associés n'est pas possible. Les exemples présentés dans la partie suivante précisent comment une affordance d'action ou d'événement peut inclure les définitions nécessaires à du raisonnement ou de la planification.
Pour ce faire, on définit de nouveaux axiomes à partir des classes définies dans l'ontologie SAREF. SAREF inclut notamment les sous-classes de `Property` suivantes : `Light` et `OnOffState`. Avec ces deux classes, on peut par exemple définir une loi physique simplifiée qui stipule que le niveau de luminosité d'une pièce dépend de la position de l'interrupteur dans la pièce.

```
(forall
  ((P st) (P l) (T o) (T o') (V v) (V v'))
```

```
(if
  (and
    (OnOffState st) (Light l)
    OBS_{o,st,v} OBS_{o',l,v'}
    (contains o o'))
  (and
    (if (= v off) (= v' dark))
    (if (= v on) (= v' average)))))
```

L'exemple ci-dessus est volontairement simplifié. Le niveau de luminosité réel d'une pièce dépend par exemple du nombre d'éclairages dans la pièce et de leur puissance individuelle.
L'ontologie SAREF définit aussi des classes de « commandes » qui permettent de rendre plus spécifique une activation sur une propriété. L'axiome suivant assigne par exemple l'une des deux classes `OnCommand` et `OffCommand` a une activation selon la valeur associée à l'activation. Par ailleurs, l'interrupteur étant un appareil passif, l'axiome définit que l'état observé de l'interrupteur résulte toujours d'une activation antérieure.

```
(forall ((P st) (T o) (V v))
  (if (and (OnOffState st) OBS_{o,st,v})
    (exists (T a)
      (and
        ACT_{a,st,v}
        (meets a o)
        (if (= v on) (OnCommand a))
        (if (= v off) (OffCommand a))))))
```

Dans notre article de 2020, on supposait qu'une commande s'applique à l'état de l'appareil qui exécute la commande. Ici, on introduit un lien explicite via `forProperty` et l'on caractérise plus précisément l'action exécutée par l'appareil (à travers la relation `forInvocation`) : il s'agit d'une activation simple. Les classes `OnCommand` et `OffCommand` sont des sous-classes de `OnOffCommand`.

```
(forall (alpha schi scho st i)
  (if
    (and
      AAF_{alpha,schi,scho} (OnOffState st)
      (forProperty alpha st)
      (forInvocation alpha i))
    (and
      (OnOffCommand i)
      ACT_{a,st,schi}))
```

Pour finir, SAREF permet aussi de décrire des événements simples, à travers la classe `EventFunction`, qui est le domaine de la relation `hasThresholdMeasurement`. De la même manière que l'on a spécifié la nature de l'action invoquée par un interrupteur, on peut spécifier le type d'événement associé à une souscription sur un capteur de luminosité, en donnant une mesure seuil.

```
(forall (alpha schs schn f o p v)
  (if
    (and
      EAF_{alpha,schs,schn}
      (hasThresholdMeasurement alpha st)
      OBS_{o,p,v}
      (forProperty alpha p)
      (forSubscription alpha s))
    (forall (T o')
      (if (and (contains s o') OBS_{o',p,v})
        N_{o',schn}))))
```

La mesure seuil est ici considérée comme une observation simple qui, lorsqu'elle se produit, déclenche une notification. L'observateur est ici le capteur. L'agent n'est qu'un observateur « par délégation » de la propriété physique.

Exemple 3. *Les formules suivantes reprennent les descriptions de l'exemple 2 et y ajoutent des « annotations » SAREF*

```
(and
  (LightSwitch switch)
  (OnOffState state)
  (forProperty aaf state))

(and
  (hasFunction lightSensor f)
  (EventFunction f)
  (hasThresholdMeasurement f t)
  (observedProperty t level)
  (hasResult t dark)
  (Light level)
  (forProperty eaf level))
```

Du fait de l'utilisation d'annotations SAREF dans les descriptions TD de l'interrupteur et du capteur d'illuminance, un agent est en mesure d'inférer les faits suivants :

- pour atteindre l'état cible $OBS_{t,level,dark}$, il est nécessaire qu'il y ait une activation `a` telle que $ACT_{a,state,off}$ (idem pour `average` et `on`);
- les affordances `paf1` et `aaf` peuvent être utilisées de manière équivalente pour activer l'interrupteur ;
- le système composé de `switch` et `levelSensor` n'est pas commandable pour l'état cible $OBS_{t,level,bright}$.

Dans le cas de l'affordance d'action caractérisée par une `OnOffCommand` comme dans le cas de l'affordance d'événement caractérisée par une `EventFunction`, l'action ou l'événement est en fait une activation ou observation simple. Les affordances sont donc redondantes avec les affordances de propriétés définies par les objets. Cependant, le cadre formel donné ici au travers de l'ontologie TD permet de caractériser des actions ou événements de complexité arbitraire, à travers un langage générique fait de relations temporelles entre observations et activations.

5 Conclusion

L'ontologie TD formalise la notion d'affordance sur le web, ce qui, à notre connaissance, n'avait encore jamais été proposé. La conception des axiomes de cette ontologie a été motivée par la vision selon laquelle des agents autonomes peuvent naviguer sur le web (sémantique) et agir de manière informée sur les ressources auxquels il ont accès. Si l'ontologie TD est une première étape vers le développement de tels agents, d'autres barrières sont à surmonter pour réaliser cette vision.

En particulier, du fait d'une représentation en logique du premier ordre des axiomes, il reste à étudier la décidabilité et la complexité de différentes tâches de raisonnement basées sur l'ontologie TD, comme par exemple la satisfaction de la spécification d'un système à partir de ses affordances afin d'en établir la commandabilité. Dans le cas où le problème général est indécidable, des pistes de validation formelles par modèles finis pourraient être explorées.

Références

[1] Information technology — common logic (cl) — a framework for a family of logic-based languages, 2018.

[2] James F. Allen and George Ferguson. Actions and events in interval temporal logic. *Journal of Logic and Computation*, 4(5) :531–579, 1994.

[3] A. Artale and E. Franconi. A temporal description logic for reasoning about actions and plans. 9 :463–506, 1998.

[4] Victor Charpenay and Sebastian Käbisch. On modeling the physical world as a collection of things : The w3c thing description ontology. In *The Semantic Web*, volume 12123, pages 599–615. Springer International Publishing, 2020.

[5] Roy Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. phdthesis, 2000.

[6] Michael Fisher, Dov M. Gabbay, and L. Vila. *Handbook of temporal reasoning in artificial intelligence*. Number v. 1 in Foundations of artificial intelligence. Elsevier, 1st ed edition, 2005.

[7] Joseph Y. Halpern and Yoav Shoham. A propositional modal logic of time intervals. *Journal of the ACM*, 38(4) :935–962, 1991.

[8] Sebastian Kaebisch, Takuki Kamiya, Michael McCool, Victor Charpenay, and Matthias Kovatsch. Web of things (WoT) thing description, 2020.

[9] Zohar Manna and Amir Pnueli. *The Temporal Logic of Reactive and Concurrent Systems : Specification*. Springer New York, 1992.

[10] Cesare Pautasso, Erik Wilde, and Rosa Alarcon. *REST : Advanced Research Topics and Practical Applications*. Springer New York, 2014.

[11] Erik Wilde. Putting things to REST, 2007.

[12] Jerzy Zabczyk. *Mathematical Control Theory*. Birkhäuser Boston, 2008.