Résumé en Français

Le volume d'informations présentes sur le Web s'accroît exponentiellement. Les utilisateurs comme les compagnies partagent de plus en plus leurs données, qui se trouvent distribuées sur les nombreuses machines qu'ils possèdent, ou via des services Web où ils stockent leurs informations sur des machines externes. En particulier, l'émergence de l'infonuagique[1] et des réseaux sociaux permet aux utilisateurs de partager encore plus de données personnelles. La multitude de services spécialisés offrant chacun une expérience différente à l'utilisateur complique énormément la gestion de ces informations et la collaboration des ces services dépasse rapidement l'expertise humaine. Les informations manipulées par les utilisateurs ont de nombreuses facettes : elles concernent des données personnelles (photos, films, musique, mails), des données sociales (annotations, recommandation, liens sociaux), la localisation des données (marque-pages), les informations de contrôle d'accès (mots de passe, clés privées), les services Web (moteur de recherche, archives), la sémantique (ontologies), la croyance et la provenance. Les tâches exécutées par les utilisateurs sont très variées : recherches par mots clefs, requêtes structurées, mise-à-jour, authentification, fouille de données et extraction de connaissances. Dans cette thèse, nous montrons que toute cette information devrait être modélisée comme un problème de gestion d'une base de connaissance distribuée. Nous soutenons aussi que datalog et ses extensions forme une base formelle sûre pour représenter ces informations et ces tâches. Ce travail fait partie du projet ERC *Webdam* [ERC13] sur les fondations de la gestion des données du Web. Le but de ce projet est de participer au développement de fondations formelles unifiées pour la gestion de données distribuées, le manque actuel de telles fondations ralentissant les progrès dans ce domaine.

Pour illustrer la problématique nous pouvons considérer *Joe* un utilisateur typique du Web. *Joe* possède un blog hébergé sur Wordpress.com sur lequel il poste des critiques de films qu'il a visionné récemment. *Joe* possède aussi un compte Facebook et Gmail pour communiquer avec ses amis, ainsi

[1] cloud computing

qu'un compte Dropbox pour stocker une partie des données qu'il souhaite partager. *Joe* aimerait automatiser une tâche qu'il effectue régulièrement manuellement. Chaque fois qu'il poste une nouvelle critique sur son blog *Joe* souhaiterait informer ses amis qu'un nouvel article est disponible et mettre à leur disposition le fichier du film qu'il vient de regarder. Cette tâche est possible à automatiser pour un programmeur en écrivant un script adhoc. Cependant *Joe* n'étant pas programmeur il est obligé de s'authentifier sur Wordpress.com, Facebook, Gmail et Dropbox pour y poster sa critique, envoyer un message à tous ses amis et envoyer le fichier du film. Nous proposons dans cette thèse un système permettant à *Joe* de continuer à utiliser les services Web qu'il affectionne tout en spécifiant à son ordinateur des tâches qu'il pourrait accomplir automatiquement.

Contributions

Les contributions de cette thèse sont les suivantes :

- Je présente *Webdamlog*, un nouveau langage à base de règles pour la gestion de données distribuées qui combine dans un cadre formel les règles déductives de datalog avec négation pour la définition des faits intentionnels et les règles actives de datalog $\neg\neg$ pour les mises-à-jour et les communications. Le modèle met un accent fort sur la dynamique et les interactions typiques du Web 2.0, principalement grâce à une nouveauté du langage *Webdamlog*, la *délégation* de règles permettant aux pairs de collaborer. Ce modèle est à la fois suffisamment puissant pour spécifier des systèmes distribués complexes et suffisamment simple pour permettre une étude formelle de la distribution, de la concurrence et de l'expressivité dans un système de pairs autonomes.

- Je présente l'implémentation du moteur d'évaluation de programmes *Webdamlog* qui étend le moteur datalog distribué avec mise à jour nommé Bud de deux manières. D'abord le moteur *Webdamlog* ajoute la possibilité d'évaluer des règles contenant des variables à la place des noms de relations et de pairs dans les règles. Puis afin de supporter la négation, *Webdamlog* permet aussi l'ajout et la suppression de règles dynamiquement, c'est à dire pendant l'exécution du programme. Enfin, je présente une technique d'optimisation basé sur la provenance pour la suppression des faits et des règles.

- Je présente l'architecture d'un pair *Webdamlog* contenant un moteur

d'évaluation *Webdamlog* et plusieurs adaptateurs[2] permettant au pair d'interagir avec les pairs non-*Webdamlog*. Je détails l'architecture et l'implémentation des lectures et mises-à-jour des faits et règles entre les adaptateurs et le moteur *Webdamlog*. La gestion des accès concurrents est basé sur le patron de conception[3] *Reactor pattern* [FMS09].

Je pense que ces contributions forment une bonne base pour résoudre les problèmes fréquemment rencontrés dans l'échange de données sur le Web, en particulier pour l'échange de données personnelles dans les réseaux sociaux.

Résumé de l'état de l'art

Cette thèse aborde deux domaines importants de l'informatique, les systèmes de données distribuées et l'inférence de connaissances.

Données distribuées Les systèmes distribués [ÖV99, AMR+11] sont des logiciels qui servent à coordonner les actions de plusieurs ordinateurs à travers l'envoi de messages. Ils sont caractérisés par les notions de consistance, de fiabilité, de disponibilité, de passage à l'échelle et d'efficacité. Dans le cas des bases de données, le système consiste en un ensemble de plusieurs bases de données, logiquement liées, distribuées sur un réseau d'ordinateurs. La distribution est transparente pour l'utilisateur : le résultat d'une requête ne dépend pas a priori du pair sur lequel elle a été posée. Sur le Web, la distribution est une composante de base de l'organisation du système. Le développement d'un langage commun, XML, et des autres standards, a facilité l'expansion des échanges. Enfin, les systèmes pairs-à-pairs, structurés ou non, représentent l'aboutissement d'importants efforts de recherche en matière de distribution dans lesquels les nœuds ont des comportements extrêmement variés et flexibles.

L'inférence La connaissance est utilisée pour décrire la sémantique des données. La connaissance représentés dans des formats lisibles par l'humain comme sur Wikipedia par exemple est difficile à traiter par les ordinateurs. Des systèmes d'intégration de la connaissance humaine en format interprétable par des machines est un sujet de recherche présenté dans [SKW07, LIJ+13]. Dans cette thèse, je m'intéresse surtout à la partie traitement des connaissances en format machine. Les fondements de l'inférence de connaissances reposent sur les bases de la logique mathématique, essentiellement des fragments de la

[2] wrappers
[3] design pattern

logique du premier ordre. Je m'intéresserai particulièrement aux systèmes déductifs de Hilbert. Ces systèmes sont basés sur des règles de Hilbert dans lesquels une règle déduit un nouveau fait à partir d'une conjonction de conditions sur un ensemble de faits déjà connus. Historiquement l'un des premiers langages à base de règles est Prolog [CR93] qui repose sur un algorithme d'évaluation nommé SLD [EK76]. Dans cette thèse je m'intéresse à datalog, un langage plus restreint que Prolog et qui offre de bonnes propriétés de terminaison. Datalog est un langage particulièrement adapté aux bases de données qui supporte nativement la récursion contrairement à SQL le langage habituellement utilisé dans les systèmes de gestion de base de données.

Le langage que nous présentons dans la section suivante est basé sur datalog [AHV95]. Des versions distribuées de datalog ont déjà été proposées [Hul89, NCW93, Hel10, GW10]. En général, un programme positif est distribué sur plusieurs pairs après une phase de compilation. Nous nous intéressons à un déploiement beaucoup plus dynamique, et nous introduisons en particulier la notion de délégation.

Le langage *Webdamlog*

La gestion d'information distribuée est un problème important, en particulier sur le Web. Des langages basés sur datalog ont donc été proposés pour le modéliser. Nous introduisons ici un nouveau modèle, dans lequel des pairs autonomes échangent des messages et des règles (délégation). Nous étudions en particulier les conséquences sur l'expressivité de la délégation. Nous proposons aussi des restrictions du langage qui garantissent sa convergence.

Considérons un pair *Alice-phone*, avec la relation *calendar* qui contient l'agenda personnel d'Alice sur son téléphone et la relation *confMembers* correspondant à la liste des membres de la conférence téléphonique. Voici des exemples de faits:

at Alice-phone:
calendar@Alice-phone(confTel, 06/12/2013, Paris, Alice-phone)
confMembers@Alice-phone(Bob, agenda, Bob-laptop)

La règle suivante ajoute les entrées relatives à la conférence téléphonique du calendrier d'Alice dans ceux des autres membres de la conférence téléphonique:

at Alice-phone:
$calendar@$peer(confTel, $date, $place, Alice-phone) :-
    calendar@Alice-phone(confTel, $date, $place, Alice-phone),
    confMembers@Alice-phone($name, $calendar, $peer)

Il faut noter que les pairs et le nom des messages sont traités comme des données. La règle génère le nouveau fait suivant :

agenda@Bob-laptop(confTel, 05/12/2013, Paris, Alice-phone)

Le fait décrit un message envoyé d'Alice-phone à Bob-laptop. Ce fait extensionnel est consommé par Bob-laptop lorsqu'il le lit. Comme dans les bases de données déductives, le modèle distingue entre faits extensionnels et faits intentionnels. Par exemple, la relation *confMembers* peut être intentionnelle et définie ainsi :

at Alice-phone:
intentionnel confMembers@Alice-phone(string, relation, peer)
confMembers@Alice-phone($name, $relation, $peer) :-
contact@Alice-phone($name, $relation, $peer),
group@Alice-phone($name, conf)

La sémantique du système est basée sur une sémantique locale, standard et sur l'échange de faits et de règles. Intuitivement, un pair donné calcule un nouvel état depuis son état courant en consommant ses faits locaux et en déduisant à partir de ces faits et de la sémantique locale les faits qu'il doit envoyer aux autres et à lui-même, ainsi que les règles qu'il doit déléguer aux autres. Un exemple de délégation est le suivant. Considérons la règle suivante:

at Bob-laptop:
confirm@$peer(confTel, $date, $place,Bob) :-
calendar@Bob-laptop(confTel, $date, $place, $peer),
checkAvailability@Bob-phone($date);

L'effet de la règle, étant donné le fait généré à l'intention de Bob-laptop, est d'installer la règle suivante sur le smartphone de Bob :

at Bob-phone:
confirm@Alice-phone(confTel, 05/12/2013,Paris,Bob) :-
checkAvailability@Bob-phone(05/12/2013);

Lorsque le smartphone de Bob exécute cette règle, en supposant que *confirm@Alice-phone* est extensionnel, il envoie le message suivant à Alice si *checkAvailability@*
Bob-phone(05/12/2013:

confirm@Alice-phone(confTel, 05/12/2013, Fontainebleau, Bob)

Si *confirm@Alice-phone* est intensionnel, c'est la règle suivante qui est envoyée:

at Alice-phone:
confirm@Alice-phone(confTel, 05/12/2013, Paris,Bob) :-

Sans rentrer dans les détails formels, il est intéressant d'étudier l'impact de la délégation sur l'expressivité du langage. En plus du langage général, noté WL, on peut distinguer deux sous-langages. Le premier, SWL, restreint la délégation aux vues. Le second, SWL, interdit complètement la délégation. Enfin, nous considérons les variantes autorisant les étiquetages temporels, notés WL^t, VWL^t et SWL^t respectivement. Les différences d'expressivité sont résumées sur la figure 1. Les inclusions sont strictes, à l'exception de celle de VWL^t dans VWL^t qui reste indéterminée.

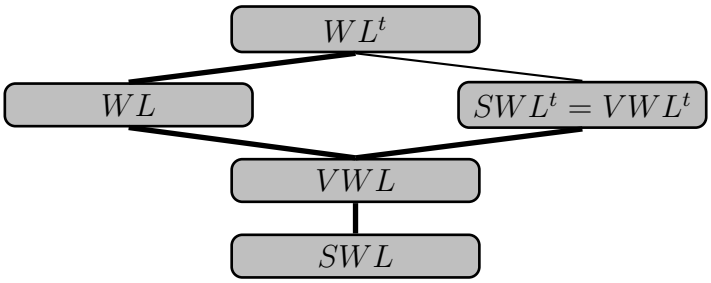

Figure 1: Expressivité des variantes de WL (les inclusions sont strictes quand l'arc est en gras)

Un autre point d'intérêt est la convergence du langage en fonction de l'ordre d'exécution des pairs. En règle générale, le résultat du calcul est a priori différent pour deux ordres d'exécution différents. Néanmoins, on peut isoler des cas monotones ou fortement stratifiés qui assurent la convergence, et ont une sémantique comparable à celle du cas où on centraliserait naturellement l'ensemble des règles et faits initiaux.

Le moteur de règles *Webdamlog*

Je considère la gestion de données distribuées sur le Web basée sur un réseau pair à pair d'acteurs autonomes et hétérogènes. Pour permettre aux pairs d'exprimer leurs propres tâches de gestion de connaissances tout en collaborant ensemble pour les tâches de gestion distribués, je propose une implémentation d'un moteur d'évaluation du langage *Webdamlog* précédemment introduit.

Le moteur *Webdamlog* s'appuie sur un moteur d'évaluation de datalog distribué nommé *Bud* [ACHM11]. Le système *Bud* supporte efficacement les mises à jour et la distribution de datalog bien qu'il n'implémente pas

la négation. *Bud* implémente l'algorithme d'évaluation semi-naïve pour l'inférence locale basée sur un système de déduction monotone positif de type chaînage avant. Afin d'évaluer un programme *Webdamlog* trois modifications majeurs au moteur ont été ajoutés:

- Le support de règles contenant des variables à la place des noms de pairs ou de relations dans les règles.
- Le support des délégations, soit la réception de règles en plus de la réception de faits.
- L'ajout de règles pendant l'exécution du système.

De plus la sémantique particulière des relations extensionnelles de *Webdamlog* qui par défaut ne sont pas permanentes impose la redéfinition des structures de mise à jour de *Bud*.

J'introduis une série d'optimisation propre l'évaluation des programmes *Webdamlog*. Premièrement une optimisation basé sur l'échange différentiel pour les délégation. Deuxièmement je montre une technique d'optimisation du type Query-Subquery [Vie86] pour les règles distribuées. Enfin je propose un technique d'optimisation plus générale pour la gestion de la suppression dans un programme datalog avec mise à jour. Cette technique basé sur la provenance de la déduction en gardant le graphe de preuves des faits et règles déduits, je propage la suppression par une mise à jour du graphe. La technique d'évaluation standard recalcule l'ensemble des relations mises à jours en relançant l'algorithme d'évaluation semi-naïve. Dans le contexte extrêmement dynamique de *Webdamlog* ou les faits et les règles changent rapidement, cette optimisation est crucial afin d'obtenir des performances raisonnable de la part du système. Je présente dans la section 4.5 une série d'expérimentation permettant de valider mes optimisations à large échelle.

L'architecture du pair *Webdamlog*

La gestion d'information sur Internet s'appuie sur une grande variété de systèmes spécialisés pour des tâches particulières. Dans l'exemple préalablement introduit, *Joe* souhaite interagir avec de nombreux services Web distants. Certains systèmes proposent des adaptateurs[4] pour intégrer les données en unique point centralisé et ainsi permettre à *Joe* de gérer automatiquement ses données via une unique interface. Je présente un pair *Webdamlog* qui muni des adaptateurs nécessaires, permet de collaborer avec les différents services

[4]wrappers

tout en gérants les données *en place*, c'est à dire en conservant la distribution des données auquel *Joe* est habitué. La nécessité de ne pas se reposer sur un unique prestataire auquel il serait nécessaire de confier toutes ses données personnelles me semble être la motivation majeur pour l'utilisation d'un système tel que *Webdamlog*.

Dans ce chapitre je décris l'architecture et l'interaction d'un pair *Webdamlog* avec les autres pairs du Web non-*Webdamlog*. L'intégration d'adaptateurs[5] autour du moteur de déduction *Webdamlog* permettent de fourni des fonctionnalités nécessaires tels qu'un interface graphique pour les interactions avec l'utilisateur, une base de données pour le stockage persistent des données et d'autre adaptateurs pour la communication par courriels[6] ou avec un réseau social comme Facebook.

Je définis un modèle général pour la gestion des évènements autre que les faits et règles *Webdamlog* basé sur le patron de conception[7] *Reactor pattern* [FMS09]. Puis je présente l'interface de programmation[8] pour les adapteurs d'un pair *Webdamlog*. Un exemple d'application réalisable grâce au système *Webdamlog* a été présenté lors d'une démonstration [4] à SIGMOD 2013. J'ai implémenté un système de réseau social pour le partage de photos lors d'une conférence. L'interface de base de cette application permet de lister ses photos, ajouter des annotations, des notes et des commentaires. Les participants étaient invités à lancer leur propre pairs avec leurs propres photos. Je leur montrais comment grâce à un petit nombre de règles *Webdamlog*, l'application permet de découvrir les autres membres de la conférence et distribuer ses photos avec ses amis. Lors de la démonstration je proposais aux participants de modifier leur pairs pour y ajouter des règles permettant de modifier le comportement de leur pairs afin de réaliser des tâches personnalisées.

Étude utilisateur

Dans cette thèse, je mets en avant l'utilisation du langage *Webdamlog*, un langage déclaratif qui permet d'abstraire les détails techniques de la distribution de la connaissances pour permettre à l'utilisateur de se concentrer sur la spécifications des tâches. Lors de cette étude, nous avons réuni un échantillons d'utilisateurs informaticiens et non informaticiens et de divers niveaux d'études pour tester leurs capacités à utiliser le langages *Webdamlog*.

[5] wrappers
[6] email
[7] design pattern
[8] Application Programming Interface (API)

Nous leur avons présenté un cours de 20 min visant à enseigner les bases du langage, puis nous leur avons fait passer un test. Les exercices du test visaient à évaluer le niveau de compréhension de petits programmes *Webdamlog* puis leurs capacités à écrire eux même des règles permettant d'accomplir automatiquement des tâches typiques qu'une application utilisant *Webdamlog* permet d'accomplir.

Conclusion

La philosophie de *Webdamlog* est de permettre de redonner le contrôle de ses données aux utilisateurs du Web. Alors que le courant actuel nous pousse à confier de plus en plus nos données à des sociétés tierces essentiellement via l'infonuagique[9], *Webdamlog* insiste sur la devise "Faites le vous même"[10], c'est à dire gérez vos propres données avec vos propres systèmes. Grâce au concept de délégation, le langage *Webdamlog* permet l'automatisation de tâches complexe de gestion de données distribuées, et en particulier celles qui requièrent la collaboration de plusieurs systèmes hétérogènes. Contrairement aux systèmes centralisées propriétaires, le code du Web est ouvert[11], et *Webdamlog* est basé sur le partage du code.

Webdamlog ouvre un grand nombre de directions de recherche. Pour conclure cette thèse, je mentionne quelques directions qui selon moi sont les plus importantes:

- Une étude utilisateur approfondie de l'utilisation de *Webdamlog* par les utilisateurs courants du Web c'est à dire ceux n'ayant que peu de connaissance de l'informatique. Il semble essentiel de comprendre les possibilités et les limitations de notre approches.

- Il serait intéressant de développer de meilleurs interfaces pour simplifier la conception d'application pour faciliter la prise en main par les futurs développeurs.

- Le contrôle d'accès pour les programmes *Webdamlog* est une pierre angulaire manquante dans notre système. Cette voie de recherche est la plus importante des priorités qui permettrai le développement de réel applications.

[9] cloud computing
[10] Do it yourself
[11] open-source

- *Webdamlog* encourage le partage de connaissances entre les pairs ou à l'intérieur d'une communauté. Certainement que de tels échanges seraient facilité par l'amélioration de *Webdamlog* avec les technologies d'ontologies du Web sémantique.

- Finalement, nous avons montré comment améliorer les performances en utilisant certaines techniques d'optimisation. Il faudrait investir plus amplement dans ce domaine pour passer à l'échelle de la toile[12], essentiellement en terme de nombre de pair, de charge de traitement et de taille des données.

[12] Web