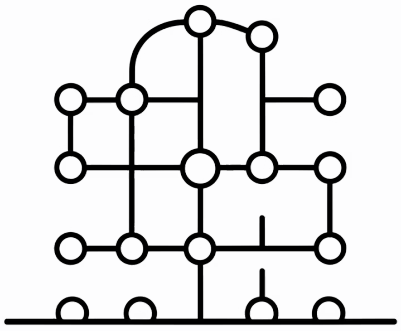
Rapport

Graphes Conceptuels Shémas et Scripts

Par Brunel Animbogo & Omar Elalaoui Fathi | **Encadré Par Mme Nidal LAMGHARI**



Ce rapport explore les graphes conceptuels, un formalisme puissant pour la représentation et la manipulation de connaissances structurées.

Développé par John Sowa en 1984, ce système révolutionnaire offre une approche unique qui combine harmonieusement logique formelle, théorie des graphes et linguistique computationnelle pour capturer la sémantique complexe du langage naturel.

Table des Matières

Introduction aux Graphes Conceptuels

Un graphe conceptuel représente un formalisme de représentation des connaissances qui s'impose comme l'un des systèmes les plus élégants et puissants dans le domaine de l'intelligence artificielle et de la gestion des connaissances. Cette approche révolutionnaire, introduite par John Sowa en 1984, combine de manière synergique trois domaines fondamentaux de l'informatique théorique et appliquée.

La **logique formelle** constitue le socle mathématique des graphes conceptuels, garantissant une base rigoureuse pour le raisonnement automatique. Plus précisément, les graphes conceptuels sont fondés sur la logique du premier ordre, ce qui leur confère une sémantique formelle précise et permet des opérations d'inférence logique fiables. Cette fondation logique assure que toute conclusion tirée à partir de graphes conceptuels est mathématiquement valide.

Les **graphes bipartis** fournissent la structure visuelle et computationnelle du système. Cette représentation graphique offre une double valeur : d'une part, elle facilite la compréhension humaine en permettant une visualisation intuitive des relations entre concepts ; d'autre part, elle permet l'application d'algorithmes de théorie des graphes pour le raisonnement automatique, notamment l'homomorphisme de graphes qui sous-tend l'opération de projection.

Enfin, la **linguistique computationnelle** permet aux graphes conceptuels de capturer fidèlement la sémantique du langage naturel. Cette dimension linguistique est cruciale car elle permet de traduire des énoncés en langage naturel vers une représentation formelle exploitable par des systèmes informatiques, tout en préservant les nuances sémantiques de l'expression originale.

Caractéristiques Fondamentales des Graphes Conceptuels

Rigoureux

Fondé sur la Logique du Premier Ordre (LPO), le système garantit une base mathématique solide et une sémantique formelle précise. Chaque graphe conceptuel peut être traduit en formule logique équivalente, permettant l'application de théorèmes de complétude et de correction. Cette rigueur formelle assure que les inférences effectuées sont valides et que le système de raisonnement est cohérent.

Intuitif

La représentation visuelle et graphique de la connaissance facilite grandement la compréhension et la communication entre experts et non-experts. Contrairement aux formules logiques abstraites, les graphes conceptuels permettent de "voir" les relations entre concepts, rendant les structures de connaissances complexes accessibles. Cette dimension visuelle favorise également la collaboration et la validation des modèles de connaissances.

Puissant

Le raisonnement automatique par Projection et Homomorphisme de graphes permet des inférences sophistiquées. L'algorithme de projection, qui recherche des correspondances entre graphes par homomorphisme, offre un mécanisme naturel et efficace pour répondre à des requêtes complexes, vérifier des hypothèses et déduire de nouvelles connaissances à partir de faits existants.

Ces trois caractéristiques se renforcent mutuellement pour créer un système de représentation des connaissances à la fois théoriquement solide et pratiquement utilisable. La rigueur formelle assure la fiabilité du raisonnement, l'intuitivité visuelle facilite la modélisation et la communication, tandis que la puissance computationnelle permet des applications pratiques dans des domaines variés comme le traitement du langage naturel, les systèmes experts ou les graphes de connaissances.

Structure et Éléments Constitutifs

Un graphe conceptuel est essentiellement un **graphe biparti**, c'est-à-dire un graphe dont l'ensemble des nœuds peut être divisé en deux sous-ensembles disjoints, et où chaque arête connecte un nœud d'un sous-ensemble à un nœud de l'autre. Cette structure bipartie impose une alternance stricte entre deux types de nœuds distincts : les concepts et les relations. Cette contrainte structurelle n'est pas arbitraire ; elle reflète la structure fondamentale du langage naturel où les entités (concepts) sont liées par des prédicats (relations).

Structure et Éléments Constitutifs (Suite)

1

Concepts [Rectangles]

Les concepts représentent les entités fondamentales du domaine de connaissance. Ils correspondent aux "choses" du monde que nous voulons modéliser : objets concrets, entités abstraites, attributs, événements, états ou processus. Dans la notation graphique, les concepts sont représentés par des **rectangles**, tandis que dans la notation linéaire CGIF, ils apparaissent entre crochets.

- **[Chat]** : Une entité animale concrète du domaine biologique
- **[Restaurant]** : Un lieu, concept spatial du domaine social
- **[Manger]** : Une action ou événement impliquant un agent et un objet
- **[Couleur: Rouge]** : Un attribut avec une valeur spécifique

Chaque concept possède un **type** (comme Chat, Restaurant, Manger) qui le positionne dans une hiérarchie ontologique, et peut avoir un **référent** qui spécifie l'instance particulière concernée. Le référent peut être individuel (*x pour une instance existentielle), spécifique (#123 pour un identifiant unique), ou générique (absence de référent).

2

Relations (Cercles)

Les relations établissent les liens sémantiques entre concepts. Elles correspondent aux prédicats du langage naturel et de la logique formelle. Dans la notation graphique, les relations sont représentées par des cercles ou ovales, tandis que dans CGIF, elles apparaissent entre parenthèses. Chaque relation possède une arité définie qui spécifie le nombre de concepts qu'elle connecte.

- (Agent) : Relation binaire désignant qui effectue l'action (arité 2)
- (Lieu) : Relation binaire indiquant où se déroule l'événement (arité 2)
- (Objet) : Relation binaire identifiant la cible ou patient de l'action (arité 2)
- (Sur) : Relation spatiale binaire entre deux entités (arité 2)

Les relations peuvent être unaires (propriétés), binaires (les plus courantes), ternaires ou d'arité supérieure. Le choix des relations et de leur arité doit refléter la sémantique naturelle du domaine modélisé et suivre les conventions établies dans l'ontologie du domaine.

La contrainte de bipartition garantit que les graphes conceptuels ont toujours une structure claire où les concepts (entités) sont reliés par des relations (prédicats), et où les relations connectent toujours des concepts, jamais d'autres relations. Cette structure facilite l'application d'algorithmes de correspondance de graphes et assure la traductibilité vers la logique du premier ordre.

Exemple :
Analyse d'une Phrase Simple

Pour illustrer concrètement la puissance expressive des graphes conceptuels, analysons en détail la phrase simple mais complète : **"Le chat est assis sur le tapis"**. Cette phrase du langage naturel, bien que courte, contient plusieurs éléments sémantiques importants qu'un graphe conceptuel peut capturer fidèlement.

Analyse Linguistique

La phrase contient :

- **Deux entités** : "le chat" et "le tapis" (noms définis)
- **Un état** : "est assis" (verbe d'état)
- **Une relation spatiale** : "sur" (préposition de localisation)

Le déterminant "le" indique des entités spécifiques mais non identifiées précisément, ce qui se traduit par l'utilisation de variables existentielles (*x, *y) dans le graphe conceptuel.

Structure du Graphe Conceptuel

Le graphe visualise clairement la relation spatiale entre les deux entités, avec la relation (Sur) agissant comme prédicat binaire connectant [Chat: *x] à [Tapis: *y].

Représentation CGIF

```
[Chat: *x] (Sur ?x ?y) [Tapis: *y]
```

Le format CGIF (Conceptual Graph Interchange Format) est le standard ISO pour échanger des graphes conceptuels entre systèmes. Cette notation linéaire capture précisément la structure du graphe :

- **[Chat: *x]** : Concept de type Chat avec référent existentiel *x
- **(Sur ?x ?y)** : Relation binaire Sur avec deux arguments
- **[Tapis: *y]** : Concept de type Tapis avec référent existentiel *y

Traduction en Logique du Premier Ordre

Ce graphe conceptuel correspond à la formule logique :

$$\exists x \exists y \, (\text{Chat}(x) \wedge \text{Tapis}(y) \wedge \text{Sur}(x, y))$$

Cette équivalence formelle garantit que le graphe possède une sémantique précise et permet l'application de techniques de raisonnement logique standard.

Cet exemple simple démontre comment les graphes conceptuels peuvent transformer une phrase du langage naturel en une représentation formelle qui préserve la sémantique tout en offrant une structure exploitable pour le raisonnement automatique. La même méthodologie s'applique à des phrases beaucoup plus complexes, avec imbrication de concepts, relations multiples et coréférences.

Les Schèmes : Modélisation de la Structure Statique

Les **schèmes** définissent la structure **statique** et compositionnelle des concepts complexes dans les graphes conceptuels. Ils répondent à la question : **"De quoi est composé X ?"** ou **"Quelles sont les parties essentielles de X ?"**

1

Rôle des Schèmes

Les schèmes définissent le **vocabulaire du domaine** et les **composants essentiels** des concepts, établissant une ontologie structurée. Ils spécifient les **contraintes structurelles** (cardinalité, types, relations) que les instances doivent satisfaire, facilitant le partage et la réutilisation des connaissances.

2

Exemple: Schème du Bus

Pour le concept de **Bus**, un schème inclut :

- **Véhicule** : Type parent (héritage).
- **Conducteur** : Composant obligatoire (cardinalité 1).
- **Sièges** : Quantité > 10.
- **Itinéraire** : Trajet prédéfini.

Ce schème capture l'essence structurelle du bus et permet sa classification.

3

Hiérarchie et Héritage

Les schèmes s'organisent en **hiérarchies de types**, reflétant les relations de subsomption. Par exemple, Bus est un sous-type de Véhicule. Cette hiérarchie permet l'**héritage de propriétés** (ex: un Bus hérite des propriétés de Véhicule). Elle est cruciale pour généraliser ou spécialiser des concepts lors de la recherche de correspondances entre graphes.

Les schèmes fournissent le vocabulaire structurel et les règles compositionnelles nécessaires pour construire des modèles de connaissances cohérents et riches, établissant l'ontologie statique du domaine.

Les Scripts :
Capture de la Dynamique Temporelle

Les **scripts** représentent le second pilier de la modélisation avancée dans les graphes conceptuels. Contrairement aux schèmes qui capturent la structure statique, les scripts modélisent la dimension **temporelle** et **processuelle** de la connaissance. Ils répondent aux questions :

"Que se passe-t-il lors de X ?", **"Quelle est la séquence typique d'événements dans la situation Y ?"**

Les Scripts (suite)

Caractéristiques des Scripts

Les scripts capturent des **séquences stéréotypées d'événements** qui se déroulent typiquement dans certaines situations. Ils incluent :

- **Relations temporelles** : (Succ) pour "succession", (Pendant), (Avant), (Après)
- **Coréférence** : Maintien de l'identité des entités à travers le temps (*x désigne la même entité dans tous les événements)
- **Rôles** : Participants récurrents dans le script (Agent, Patient, Instrument)
- **Préconditions et postconditions** : États requis avant et après le script

Script Classique : Dîner au Restaurant

Considérons le script bien connu d'un repas au restaurant, décomposé en quatre étapes principales reliées par la relation temporelle **(Succ)** pour "Succession" :

01

Entrer

```
[Client: *x] – (Agent) – [Entrer]
[Client: *x] – (Destination) – [Restaurant: *r]
```

Le client entre dans le restaurant

02

Commander

```
[Client: *x] – (Agent) – [Commander]
[Plat: *p] – (Objet) – [Commander]
```

Le client passe commande d'un plat

03

Manger

```
[Client: *x] – (Agent) – [Manger]
[Plat: *p] – (Objet) – [Manger]
```

Le client consomme le plat commandé

04

Payer

```
[Client: *x] – (Agent) – [Payer]
[Addition] – (Objet) – [Payer]
```

Le client règle l'addition

Format Standard CGIF et Formalisation

Le **CGIF** (Conceptual Graph Interchange Format) est le format d'échange standardisé (ISO 24707:2007) pour les graphes conceptuels, assurant l'interopérabilité et le partage des connaissances.

Variables et Référents

Le CGIF utilise des marqueurs pour les instances :

- ***x, *y, *z** : Variables existentielles
- **?x, ?y** : Variables de relation
- **#123** : Identifiants de coréférence
- **Absence de marqueur** : Concept générique

Syntaxe des Concepts

Structure : **[Type: référent]**

- **[Chat]** : Concept générique
- **[Chat: *x]** : Chat existentiel
- **[Couleur: Gris]** : Valeur d'attribut
- **[Personne: #Marie]** : Instance nommée

Syntaxe des Relations

Structure : **(Nom ?arg1 ?arg2 ... ?argN)**

- **(Sur ?x ?z)** : Relation binaire
- **(Agent ?a)** : Relation unaire
- **(Entre ?x ?y ?z)** : Relation ternaire

Exemple Complet

```
[Chat: *x] (Sur ?x ?z) [Tapis: *z]
[Chat: *x] (Couleur ?x ?c) [Couleur: Gris]
```

Cette représentation CGIF signifie : "Il existe un chat (*x) sur un tapis (*z), et ce chat (*x) est de couleur grise." La coréférence garantit l'identité des entités.

Traduction en Logique

Correspond à la formule logique du premier ordre :

$$\exists x \exists z \exists c\, (\text{Chat}(x) \wedge \text{Tapis}(z) \wedge \text{Sur}(x,z) \wedge \text{Couleur}(c) \wedge c=\text{Gris} \wedge \text{ACouleur}(x,c))$$

Cela démontre une sémantique précise et la compatibilité avec le raisonnement logique.

Avantages du CGIF

- **Lisibilité** : Plus proche du langage naturel
- **Standardisation** : Norme ISO pour l'interopérabilité
- **Expressivité** : Couvre toutes les constructions CG
- **Traçabilité** : Lien clair avec la représentation graphique

Le CGIF facilite le stockage, la transmission et le traitement automatique des graphes conceptuels de manière standardisée.

Raisonnement par Projection et Homomorphisme

La **projection** est l'algorithme fondamental du raisonnement dans les systèmes de graphes conceptuels. C'est un **homomorphisme de graphes** qui vérifie si un patron (requête, règle) correspond à des faits dans une base de connaissances.

Formellement, une projection est une fonction π d'un graphe G_1 (patron) vers un graphe G_2 (faits) qui préserve la structure :

- Chaque concept de G_1 est mappé vers un concept de G_2 (respectant la hiérarchie de types).
- Chaque relation de G_1 est mappée vers une relation de même nom dans G_2.
- La structure des arcs est préservée.

Conditions sur les Concepts

Pour projeter C_1 (patron) sur C_2 (fait) :

- **Contrainte de type** : $Type(C_1) \leq Type(C_2)$ (ex: [Animal] → [Chat]).
- **Contrainte de référent** : Référent de C_1 générique ou identique à C_2.

Ceci permet à un patron général de matcher des faits spécifiques.

Conditions sur les Relations

Pour les relations, les contraintes sont plus strictes :

- **Nom identique** : Même nom pour la relation du patron et du fait.
- **Arité identique** : Même nombre d'arguments.
- **Préservation de structure** : Les connexions entre concepts et relations doivent être préservées.

Les relations n'ont généralement pas de hiérarchie de subsomption.

Applications Pratiques de la Projection

Requêtes

Formule des requêtes sur une base de connaissances. Le graphe de requête trouve les instances correspondantes.

Exemple : "Qui mange quoi ?" → [Personne: ?x] (Agent) [Manger] (Objet) [Nourriture: ?y]

Inférence

Les règles sont des graphes (prémisse/conclusion). Si la prémisse se projette, la conclusion peut être ajoutée.

Exemple : Si [Animal: *x] (Sur) [Tapis] alors [Tapis] (État) [Sale]

Validation

Vérifie si une situation est une instance d'un script ou schème.

Exemple : Vérifier qu'une séquence d'actions correspond à un "dîner au restaurant".

La projection est une opération de raisonnement centrale pour la reconnaissance de patterns, la recherche d'information et l'application de règles. Bien que NP-complète, sa base formelle garantit des propriétés algorithmiques solides.

Outils, Implémentations et Perspectives

L'écosystème des graphes conceptuels s'est développé au fil des décennies avec plusieurs implémentations matures, principalement en Java pour des raisons de robustesse, performance et typage fort. Cependant, des alternatives en d'autres langages existent également, offrant flexibilité et intégration avec des écosystèmes modernes de data science et d'intelligence artificielle.

CoGui (Java)

Interface visuelle intuitive pour créer et manipuler des graphes conceptuels de manière interactive. CoGui offre un éditeur graphique WYSIWYG permettant de dessiner directement les graphes, avec export natif en CGIF et XML. Particulièrement recommandé pour l'enseignement, le prototypage rapide et l'apprentissage des concepts fondamentaux des graphes conceptuels.

Amine (Java)

Plateforme complète et ambitieuse pour le développement de **systèmes multi-agents** basés sur les graphes conceptuels. Amine offre un support avancé des scripts, des règles d'inférence et de la projection. C'est un écosystème complet pour construire des systèmes intelligents complexes intégrant représentation des connaissances, raisonnement et agents autonomes.

Notio (Java)

API robuste et complète pour la manipulation programmatique native des graphes conceptuels. Notio fournit des classes Java pour tous les éléments du formalisme (concepts, relations, graphes, projection) et est idéal pour l'intégration de fonctionnalités de graphes conceptuels dans des applications backend de production nécessitant fiabilité et performance.

Python + NetworkX

Alternative flexible combinant NetworkX (bibliothèque de graphes) avec développements personnalisés. Offre plus de **liberté et d'intégration** avec l'écosystème Python moderne (NumPy, Pandas, scikit-learn, deep learning), mais nécessite implémentation DIY des algorithmes spécifiques aux graphes conceptuels comme la projection avec contraintes de types.

Applications et Perspectives

Pourquoi Java Domine l'Écosystème des Graphes Conceptuels ?

1 Écosystème Natif et Maturité

Les outils fondateurs (CoGui, Amine, Notio) ont été conçus spécifiquement pour les graphes conceptuels dès les années 1990-2000, bénéficiant de décennies de développement et de raffinement. Ils implémentent fidèlement le formalisme complet de Sowa.

2 Typage Fort et Sécurité

Le système de types de Java garantit la cohérence ontologique à la compilation, détectant les erreurs de types avant l'exécution. Ceci est crucial pour les systèmes de représentation des connaissances où la validité des types est fondamentale.

3 Performance des Algorithmes de Projection

Les algorithmes d'homomorphisme de graphes avec contraintes de types (projection) sont hautement optimisés en Java, bénéficiant du JIT compiler et de structures de données efficaces. La performance est critique car la projection est NP-complète.

4 Standardisation CGIF et Interopérabilité

Support natif complet du format d'échange ISO CGIF, facilitant l'interopérabilité entre outils et le partage de bases de connaissances. Les parsers et générateurs CGIF sont matures et conformes aux standards.

Applications Pratiques et Domaines d'Application

NLP	Parsing syntaxique et traduction sémantique	Capture fidèle de la sémantique du langage naturel, support de l'ambiguïté
Knowledge Graphs	Ontologies et graphes de connaissances	Hiérarchies de types, schèmes structurels, interopérabilité CGIF
Systèmes Experts	Bases de connaissances intelligentes	Raisonnement par projection, règles d'inférence, scripts procéduraux
Recherche d'Information	Recherche sémantique avancée	Requêtes structurées, correspondance partielle, généralisation/spécialisation
Web Sémantique	Annotation et intégration de données	Expressivité supérieure à RDF, traductibilité en OWL, standards ISO

Conclusion

Les graphes conceptuels continuent d'évoluer avec l'émergence de nouvelles applications en intelligence artificielle, notamment dans la fusion avec les approches neuronales (Graph Neural Networks sur graphes conceptuels) et dans les architectures hybrides combinant raisonnement symbolique et apprentissage automatique. L'avenir du domaine réside probablement dans ces approches neuro-symboliques exploitant le meilleur des deux paradigmes.