Des modèles au temps réel: le système KSE de traitement des alarmes nucléaires

Jean-Luc Dormoy
DER-EDF IMA/TIEM
1, avenue du Général de Gaulle 92141 Clamart Cedex FRANCE
Email: Jean-Luc.Dormoy@der.edf.fr

François Cherriaux & Jean Ancelin
DER-EDF EP/CCC
6, quai Watier 78401 Chatou Cedex FRANCE

Résumé: Ce papier présente essentiellement une application, à savoir un système de surveillance en temps-réel basé sur les modèles pour les centrales nucléaires. Ce système a été installé dans une centrale en France. Nous décrivons comment nous avons utilisé des techniques d'IA variées pour le construire : une approche basée sur les modèles, un modèle logique de son fonctionnement, une mise en œuvre déclarative de ces modèles, et des techniques originales de compilation de connaissances visant à automatiquement générer le système expert temps-réel à partir de ces modèles. Certaines de ces techniques ont simplement été empruntées à la littérature, mais nous avons dû en modifier certaines, voire en inventer de nouvelles.

Nous avons été poussés à ces innovations par la nécessité de construire un système véritablement opérationnel. Il nous semble que, considérant l'état de l'art dans la littérature de l'IA, il reste une somme de travail importante à consacrer à la "théorie" - même si elle reste d'une portée modeste - et à l'intégration d'approches diverses lorsque l'on ambitionne une application de grande taille. Nous pensons que cela révèle un besoin, qui ne semble pas rempli aujourd'hui, et qui devrait pousser à rediriger les efforts pour les "théoriciens de l'IA appliquée".

Mots-clés: Applications, raisonnement basé sur les modèles, systèmes experts temps-réel, compilation de connaissances.

1 Introduction

Ce papier présente l'architecture du système KSE, qui vise à expliquer les causes d'événements indésirables dans une centrale nucléaire, et à fournir à ses opérateurs une description de l'état opérationnel de l'installation. En particulier, nous décrivons une technique originale de compilation de connaissances, qui rend possible de générer automatiquement un système expert temps-réel à partir de modèles de l'installation et d'une spécification logique de la tâche à effectuer. Nous discutons aussi deux attributs des applications temps-réel, qui méritent à notre avis plus d'attention : la taille, et les erreurs.

Le paragraphe 2 décrit les buts du système. Le paragraphe 3 décrit son architecture. Nous y montrons tout d'abord comment la centrale, ses comportements physiques et fonctionnels, et un modèle logique de la tâche à accomplir sont mis en œuvre. Nous présentons ensuite la stratégie de compilation de connaissances que nous utilisons pour compiler ces modèles. Le paragraphe 4 discute de questions diverses autour de KSE, en particulier concernant la justesse d'un système en relation avec la justesse de son modèle. Nous mentionnons dans le paragraphe 5 les principaux problèmes rencontrés dans le développement de KSE, à savoir sa grande taille, et les erreurs présentes dans les modèles.

2 Buts généraux du système KSE

2.1 Le problème

Le projet KSE a visé à développer un système temps-réel pour le traitement des alarmes dans les centrales nucléaires. Cela signifie qu'à chaque fois qu'une alarme se déclenche, le système doit être capable de fournir à l'opérateur la cause initiale de l'alarme, et une description actualisée de l'état de l'installation en termes de disponibilité fonctionnelle de ses composants.

Cela n'est pas un problème de diagnostic, dans le sens "pur" donné dans la littérature IA. Ici, un "mauvais comportement" ne signifie pas qu'un composant est "cassé" - bien que cela soit aussi possible - mais simplement que quelque chose arrive qui rend la conduite de l'installation plus contrainte. Ainsi, les mauvais comportements sont exprimés en termes d'états physiques, et le système doit découvrir la cause première des mauvais comportements courants.

Une autre différence est qu'il n'y a pas de manque d'information dans les centrales nucléaires. Au contraire, les opérateurs sont quelquefois noyés sous un flot d'informations. Jusqu'à 300 alarmes peuvent apparaître en une seule minute, tout en n'ayant qu'un petit nombre de causes.

2.2 Ce que le système KSE est

Le système que nous avons développé a été en test pendant plus de deux ans dans une centrale en France. Il est entièrement automatique, c'est-à-dire qu'il prend ses données du système d'information de la centrale, et présente son diagnostic à l'opérateur sans son intervention.

En fait, le système temps-réel est un système expert, mais nous ne l'avons pas écrit. Au lieu de cela, nous avons développé un outil basé sur les modèles, où des modèles de natures diverses sont mis en œuvre, et un compilateur de connaissances, qui génère automatiquement le système temps-réel à partir de son modèle. L'intention initiale était de baser KSE sur des modèles d'une manière aussi déclarative que possible d'un côté, et d'avoir un système temps-réel efficace d'un autre. Il est

bien connu que ces objectifs sont antagonistes lorsque pris naïvement. Notre solution a donc été de développer un composant de compilation de connaissances.

2.3 Ce que le système KSE n'est pas

La champ d'application de notre système a été limité à la partie électrique de l'installation. En fait, tous les composants de la centrale électriquement connectés sont concernés, mais les processus thermohydrauliques n'ont pas été pris en compte. En effet, résoudre le problème pour toute la centrale aurait été infaisable dans une première étape. La conséquence de ces choix est que le système n'a pas besoin de raisonner sur le temps (une vue instantanée de l'installation est suffisante), et que nous n'avons pas eu à manipuler des modèles physiques sophistiqués (les processus thermodynamiques).

Ainsi, comme nous le verrons, le système KSE n'est pas l'application de techniques d'IA très sophistiquées. Il est plutôt le résultat de l'application et de l'intégration de techniques relativement complexes, et relativement nouvelles. Nous expliquons au § 5 les deux raisons pour lesquelles il en est ainsi : la taille, et les erreurs.

3 Le système KSE : Construire un modèle, et le compiler

Ce paragraphe est divisé en 3 parties. Nous décrivons d'abord le modèle de la centrale utilisé, puis celui de la logique de fonctionnement du système expert temps réel. Nous montrons ensuite comment ce modèle peut être compilé en un système temps réel.

3.1 Modèles de l'installation et de la tâche de surveillance

Notre volonté constante pour KSE a été de séparer les diverses sources de connaissances de l'application. Nous avons essentiellement ici trois grands morceaux. Premièrement, il y a un modèle de la centrale, c'est-à-dire de ce dont elle est constituée, de la nature et des relations entre ses composants, et d'autres informations utiles. Ce modèle s'adapte bien aux cadres orienté-objets ou des réseaux sémantiques.

Deuxièmement, il y a une description des relations causales entre les diverses quantités véhiculées par l'installation. Ces quantités peuvent être d'une nature physique ou fonctionnelle. Cette description prend la forme de principes exprimés comme des "implications causales" en logique des prédicats du premier ordre. En fait, seule une partie du calcul des prédicats a été utilisée, qui d'un côté est suffisante pour exprimer les principes, et de l'autre rend possible une compilation efficace du système.

Troisièmement, nous avons un modèle logique de l'action désirée du système expert temps-réel. Bien qu'il soit relativement simple, déclarer ce modèle était nécessaire pour remplir notre objectif de séparation des sources de connaissances. Chaque source a son propre mécanisme de vérification.

Les première et seconde sources - description de la centrale et de son comportement - sont probablement partageables avec d'autres applications. La troisième source lui est particulière.

3.1.1 Description de la centrale

Des informations de diverses natures sont décrites ici.

Premièrement, il y a une classification hiérarchique des composants et des sortes de composants. Un objet (classe ou composant) peut être une instance de plusieurs classes. Un mécanisme d'héritage multiple a été mis en œuvre, qui traite des exceptions. Cela signifie que toute propriété peut être héritée par un objet à partir de sa classe (règle par défaut), sauf quand une exception a été explicitement énoncée. Ainsi, certaines classes ou certains objets ont des propriétés exceptionnelles.

Deuxièmement, les liens entre composants sont décrits, de façon évidente. Il y a plusieurs sortes de liens selon la nature du fluide qu'ils transportent, par exemple l'électricité ou la disponibilité. Les liens sont causalement orientés dans les directions possibles de circulation du fluide.

Troisièmement, il y a un modèle des attributs des objets. Celui-ci décrit essentiellement les valeurs possibles des attributs, et leurs exclusions mutuelles. Les attributs peuvent décrire des positions de composants (ouvert/fermé), des valeurs de quantités (0/1 ou bas/moyen/haut), les types d'informations fournis par l'instrumentation de la centrale, les états indésirables, quels états peuvent être considérés comme explication d'un état indésirable, etc. Le genre de connaissances représenté ici est nécessaire à la résolution de la tâche de surveillance, mais aussi à la compilation des modèles.

Le problème principal ici n'est pas la théorie - tout cela est relativement classique - mais la taille. Le système KSE travaille sur 12000 composants, et leur modèle se monte à 150000 triplets objet-attribut-valeur. Une tentative a été conduite pour retrouver cette connaissance à partir de bases de données CAO. Cependant, certaines propriétés n'étaient pas décrites, telles que la nature des fluides et les attributs correspondants. La plus grande partie de cette "base de données" a donc été reconstruite manuellement pour la centrale où le système a été testé. Un composant de vérification de cohérence a été ajouté, qui vérifie la base de données, mais l'expérience a montré que des erreurs restaient. Cela est un problème difficile que nous discuterons au § 4.

3.1.2 Description comportementale et fonctionnelle

Nous décrivons ici les relations causales entre quantités relatives à des composants de types donnés reliés entre eux. Prenons un exemple simple. Considérons des matériels électriques - causalement - connectés à un tableau électrique (figure 1).

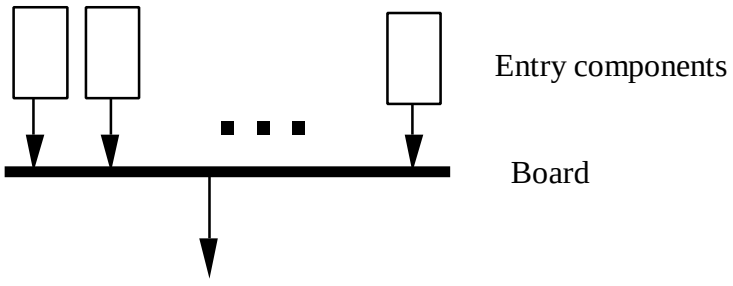

Figure 1: n composants connectés à un tableau électrique

Les principes physiques correspondants sont les suivants :

[x Ako TableauElectrique
∃y[y Ako MaterielElectrique
y Alimente x

[x Ako TableauElectrique
∀y [y Ako MaterielElectrique
y Alimente x]

y Potentiel 1]
]
cause-->
[x Potentiel 1]

==> [y Potentiel 1]
]
cause-->
[x Potentiel 1]

Les mots Ako, MaterielElectrique, Potentiel, etc, font référence à des entités du modèle de la centrale.

Nous avons ici deux modèles causaux. En fait, ils pourraient ici être déduits l'un de l'autre, mais ce n'est pas le cas général. Le fait que les lois physiques sont causalement orientées introduit implicitement des aspects fonctionnels : pour la classe de composants décrite par un principe donné, les fluides coulent dans une certaine direction. C'est une information globale, qui ne peut être connue autrement, sinon à considérer la connectivité toute entière (en particulier les sources de fluide). En fait, nous avons précisément voulu séparer les principes de la description de la centrale. Néanmoins, si la causalité était supprimée, les principes fusionneraient en une équivalence unique.

Les deux principes ci-dessus mentionnent une quantité physique (le potentiel). D'autres principes mentionnent d'autres sortes de quantités, comme la disponibilité - c'est-à-dire la possibilité pour l'opérateur d'utiliser le composant. Les propriétés fonctionnelles que nous avions à traiter avaient l'heureuse caractéristique de pouvoir être spécifiées localement, et donc en utilisant des principes.

Les principes mentionnent des classes. Ils peuvent être hérités le long de la classification, avec des exceptions.

Ces principes sont simples. Certains sont plus compliqués, mais il s'est avéré qu'ils ont tous une forme commune. En fait, ils mentionnent tous un certain x d'une certaine classe de matériels, puis d'autres y d'autres classes, et ce quantifiés soit universellement (et alors avec une implication), soit existentiellement. Il n'y a pas de quantificateurs imbriqués. De plus, la forme "pour tout y différent de y_0 …" est souvent utilisée. En conséquence, nous avons imposé d'exprimer les principes dans ce langage des prédicats simplifié, et nous avons introduit deux quantificateurs modifiés, "pour tout … sauf …" et "il existe … différent de …" pour leur aspect pratique. Tous les programmes traitant cette base de connaissances ont été limités à ce "calcul des prédicats restreint".

Un sous-système vérifie les principes, qui regarde s'ils ne sont pas complètement stupides. Une erreur non immédiatement détectée dans les principes est arrivée une seule fois.

Il y a environ 250 principes causaux.

3.1.3 Modèle de la tâche

La tâche de KSE consiste à expliquer les causes initiales des états non désirés. En fait, l'algorithme est relativement simple. A partir d'un effet, on peut générer toutes les causes potentielles de cet effet en utilisant les principes. Si un composant a un état connu ou déduit à partir des instruments, et qui est contradictoire à une des hypothèses, alors cette hypothèse peut être écartée. En définitive, toutes les hypothèses restantes sont des causes (potentielles) de l'effet indésirable.

Ainsi, la tâche de KSE peut être divisée en trois parties : déduire une image aussi complète que possible de l'état de la centrale (en termes de valeurs de quantités physiques) à partir des données de l'instrumentation, puis générer les hypothèses possibles, et enfin éliminer les hypothèses incohérentes. Chacune de ces étapes utilise les principes causaux.

Ce comportement peut être décrit par six règles d'inférence qui utilisent deux symboles supplémentaires, à savoir "$\propto$" pour "est une cause potentielle de" (hypothèse) et "∂" pour "est une cause de" (définitivement) :

A cause B et A $\rightarrow$ B	*Déduction causale*
A cause B et $\neg$B $\rightarrow \neg$A	*Contraposition*
A cause B et B et $\propto$B $\rightarrow \propto$A	*Génération d'hypothèse*
A cause B et A $\rightarrow \neg\partial$B	*Négation causale d'hypothèse*
$\neg$A $\rightarrow \neg\partial$A	*Négation d'hypothèse par état*
$\propto$A $\rightarrow \partial$A [Défaut]	*Confirmation d'hypothèse*

La dernière règle est une règle de défaut. Cet ensemble de règles d'inférence doit démarrer avec les hypothèses (de la forme $\propto$H) correspondant aux états indésirables observés. Il est clair que l'état de la centrale est calculé par les deux premières règles, que les hypothèses sont générées par la troisième, écartées par les quatrième et cinquième, et que les causes initiales sont établies par la sixième.

Le défaut est ici relativement simple. Pour obtenir un ensemble minimal de causes initiales, nous devons juste appliquer les cinq premières règles jusqu'à saturation, puis la règle de défaut. De plus, les deux premières règles peuvent être utilisées en premier, puis abandonnées, et la cinquième utilisée avant la quatrième.

Contrairement à tous les autres aspects de KSE, cette spécification de la tâche du système temps-réel n'a pas été incorporée au système. Il s'agit seulement d'une formalisation pour ses concepteurs. Nous verrons les conséquences de cela dans la partie sur la compilation.

Remarque 1 : Il y a d'autres règles d'inférence qui seraient cohérentes avec la sémantique désirée de $\propto$ et ∂, par exemple $\neg\propto$A $\rightarrow \neg\partial$A. Elles sont cependant inutiles dans notre contexte.

Remarque 2 : La règle de génération d'hypothèse mentionne B dans sa partie gauche. Cela n'est pas indispensable, mais généraliser cette règle en supprimant B conduirait à générer beaucoup plus d'hypothèses. Inclure B signifie "nous avons trouvé des causes initiales aux événements indésirables, bien qu'il pourrait en exister de "plus initiales". Mais, vue notre connaissance, nous n'avons aucun moyen de les prouver". Cela revient à une sorte *d'ignorance épistémique*.

3.2 Compilation des modèles

Une fois les modèles entrés dans l'ordinateur, un autre programme les prend en donnée et les compile en le système temps-réel. Avant de décrire comment cela est effectué, nous donnons une idée de ce en quoi consiste le système temps-réel.

3.2.1 La cible : le système temps-réel

Le système temps-réel travaille comme une boucle infinie. A chaque étape de la boucle, il prend les données issues de l'instrumentation de la centrale (essentiellement les valeurs de quantités physiques). Il effectue alors ses calculs comme décrit ci-dessus, et fournit à l'utilisateur les causes initiales des alarmes et des états indésirés (s'il y en a), et "l'état fonctionnel" de la centrale, par exemple la disponibilité des composants. Puis, un nouveau parcours de la boucle est effectué.

Ainsi, le système temps-réel ne travaille qu'avec des informations temps-réel (!). Toutes les connaissances sur l'installation (les composants et leurs liens, les principes) y sont "câblés". Le système temps-réel est constitué de règles de production d'ordre 0^+ réparties en trois grands groupes correspondant aux trois sous-tâches : déduire l'état de l'installation, générer les hypothèses, et les éliminer. Deuxièmement, chaque règle est spécifique à un composant, et met en œuvre une de ces activités pour ce composant. Par exemple, pour un tableau électrique particulier b112 ayant trois composants connectés en entrée, disons e1, e2 et e3, le système expert contient les quatre règles :

si e1_a_pour_potentiel 1
alors b112_a_pour_potentiel 1 *(deux autres règles)*

si e1_a_pour_potentiel 0
et e2_a_pour_potentiel 0
et e3_a_pour_potentiel 0
alors b112_a_pour_potentiel 0

Ces règles représentent l'application au tableau b112 des principes du § 3.1.2 selon la première règle d'inférence.

Cet exemple donne une idée du processus de compilation. Les règles d'inférence logiques sont instanciées par les principes, et appliquées à la description de l'installation, ce qui conduit à des règles d'ordre 0^+. Autrement dit, tout ce qui n'est pas de nature temps-réel dans les modèles est "oublié" ou "câblé" dans les règles temps-réel. Il ne reste que de l'information temps-réel, qui ne peut évidemment pas être connue au moment de la compilation.

Dans KSE, cela aboutit à environ 47000 règles.

3.2.2 Stratégie de compilation des connaissances

La compilation est effectuée en trois étapes.

La première étape consiste à "mélanger" les principes et le modèle opérationnel (i.e. les règles d'inférence). Cette étape produira de nouveaux "principes", qui ne sont pas causaux, mais qui décrivent les déductions pouvant être effectuées sur une centrale générique en utilisant les règles d'inférence. Pour résumer, la première étape instancie les règles d'inférence par les principes. Les nouveaux "principes" ont la même forme que les principes causaux. Nous les appellerons principes inférentiels. Cette étape n'utilise qu'une petite partie du modèle de la centrale, à savoir la description des attributs et de leurs valeurs possibles.

La seconde étape part des principes disponibles (causaux et inférentiels) et les transforme en un ensemble de règles de production du premier ordre qui n'utilisent aucun quantificateur explicite. Cette étape existe pour deux raisons. Premièrement, nous ne disposions pas d'un langage de production qui accepte les quantificateurs explicites, et en particulier pas nos quantificateurs modifiés. Deuxièmement, ces règles ont une forme très particulière. Dans leurs parties gauches, elles mentionnent des relations sur les objets qui sont connues dans le modèle de la centrale (par exemple les relations de lien entre composants). Dans leur parties droites, elles écrivent une règle d'ordre 0^+ conformément à un schéma, dont les variables seront instanciées par les données de l'installation réelle lorsque les productions seront exécutées. Cette seconde étape utilise, outre les principes de tous ordres, la classification hiérarchique contenue par le modèle de la centrale.

La troisième étape consiste à exécuter l'ensemble de productions du premier ordre précédent sur la mémoire de travail contenant la description complète de la centrale (composants, liens, etc), et produire ainsi l'ensemble de règles d'ordre 0+ qui est le système temps-réel. Cette troisième étape utilise la plus grosse partie du modèle de la centrale, au moins en taille, c'est-à-dire la description de la centrale jusqu'à chacun de ses composants.

3.2.3 Détail de la compilation des connaissances : des principes causaux aux principes inférentiels

Nous décrivons ici certains aspects de la première étape mentionnée ci-dessus. Les seconde et troisième étapes sont décrites dans les sous-paragraphes suivants.

Seules les quatre premières règles interviennent dans la transformation des principes. Chaque principe est transformé par chaque règle d'inférence selon une grammaire. Cette grammaire établit comment transformer une condition ou une conclusion causale en une nouvelle condition ou conclusion. Cette grammaire est triviale - elle se résume à l'identité - pour la règle de déduction causale, qui laisse les principes intacts. Elle doit par contre être détaillée pour les autres règles d'inférence.

- **Contraposition :** Les principaux problèmes sont de transformer les quantificateurs, et l'interprétation de la négation. Le point principal est que nous voulons construire des principes non-causaux qui soient des *clauses de Horn*.

  Pour expliquer cela, prenons l'exemple d'un principe causal ayant la forme :

  "si x est dans la classe C tel que,
  pour tout y de la classe C' lié à x par L(x,y), K(y) est vrai
  alors RHS(x)"

  Lorsqu'une variable x est instanciée par un objet spécifique X via la descrition de la centrale, il n'y a qu'un nombre fini d'instances pour y rendant vraie la formule prédicative "y dans la classe C lié à x par L(x,y)", disons $Y_1,\ldots,Y_p$. Ainsi, le principe instancié est équivalent à

  "si $K(Y_1),\ldots,K(Y_p)$ alors RHS(X)"

  Maintenant, cette clause peut être transformée par transposition pour tout i de [1,p] en les clauses de Horn suivantes :

  "si $\neg$RHS(X) et $K(Y_1),\ldots,K(Y_{i-1}),K(Y_{i+1}),\ldots,K(Y_p)$ alors $\neg K(Y_i)$"

  Si nous revenons aux principes, ces clauses sont l'instanciation du principe contraposé recherché :

  "si x est dans la classe C tel que $\neg$RHS(x) et,
  il existe y_0 dans la classe C' lié à x par $L(x,y_0)$ tel que
  pour tout y de la classe C' différent de y_0 et lié à x par L(x,y), K(y) est vrai
  alors $\neg K(y_0)$"

Par exemple, le principe "si toutes les entrées d'un tableau électrique ont pour Potentiel 0, alors le tableau a pour Potentiel 0" est contraposé en "si un tableau électrique a pour Potentiel 1, et si toutes les entrées sauf e_0 ont pour Potentiel 0, alors e_0 a pour Potentiel 1"

Examinons maintenant un autre problème : l'interprétation de la négation. Potentiel est un attribut ayant exactement deux valeurs possibles mutuellement exclusives (0 et 1), et cela est mentionné dans le modèle de la centrale. Ainsi, dans ce cas, le système interprète "x Potentiel 1" comme la négation de "x Potentiel 0", et réciproquement. Cependant, un attribut peut avoir plus de deux valeurs mutuellement exclusives. Dans ce cas, le système interprète "¬(x Attr Val)" comme "x ne peut pas avoir la valeur Val pour l'attribut Attr". Pour mettre en œuvre cela, le système synthétise dans ce cas un nouvel attribut Imp_Attr, qui doit se lire "valeur impossible pour Attr".

Si à un moment donné il est possible de déduire que x.Attr ne peut avoir aucune de ses valeurs possibles sauf v, alors v doit être affectée comme valeur de x.Attr. Cela ne fait pas partie de la transformation des principes, mais d'une étape postérieure du processus de compilation. Pour chaque attribut ayant plus de deux valeurs possibles, et pour tout composant concerné par cet attribut, des règles seront ajoutées au système temps-réel qui feront ce genre de déduction. Fort heureusement, il ya peu de tels attributs en pratique. Le système temps-réel n'est donc pas noyé sous ce genre de règle stupide.

Nous ne décrivons pas maintenant complètement la grammaire de la contraposition, mais nous donnons quelques indications. Les règles de grammaire transforment localement les conditions et les conclusions causales en conditions et conclusions contraposées. Voici par exemple deux règles de transformation de la grammaire :

($\exists y \in C'$ L(x,y) et K(y)) est une condition causale ---->
($\forall y \in C'$ L(x,y)) est une condition contraposée *et*
(¬K(y)) est une conclusion contraposée

($\forall y \in C'$ L(x,y) ==> K(y)) est une condition causale ---->
($y_0 \in C'$, $\forall y \in C'-\{y_0\}$ L(x,y) ==> K(y)) est une condition contraposée *et*
(¬K(y_0)) est une conclusion contraposée

x est ici une variable libre par rapport aux formules citées, L se réfère à la description structurelle de la centrale, et K à son état (seul K représente de l'information temps-réel).

Génération d'hypothèses : Les buts sont les mêmes que pour la contraposition, c'est-à-dire obtenir des principes qui s'instancient en clauses de Horn, mais les moyens en sont quelque peu différents. En vérité, cette règle a été mise en œuvre sous une forme affaiblie lorsqu'un quantificateur universel du principe causal est transformé. Si nous considérons un principe causal ayant la forme :

"si x est dans la classe C tel que,
pour tout y de la classe C' lié à x par L(x,y), K(y) est vrai
alors RHS(x)"

ce principe est transformé en le principe de génération d'hypothèses suivant :

"si x est dans la classe C tel que RHS(x),

si ∝RHS(x)
si y de la classe C' est lié à x par L(x,y)
alors ∝K(y)"

Une forme non affaiblie serait :

"si x est dans la classe C tel que RHS(x),
si ∝RHS(x)
alors ∝[∀y∈ C' L(x,y) ==> K(y)]"

Cela tient aussi si nous substituons un quantificateur existentiel au quantificateur universel. Dans ce cas, le principe de génération d'hypothèses déduira qu'il y a plusieurs causes potentielles, mais il "oubliera" qu'une seule de ces causes est nécessaire pour expliquer les effets.

Cela signifie que le processus de génération d'hypothèses ne garde pas trace des "et" et des "ou" des causes. En pratique, cela n'est pas un problème, parce qu'en définitive toutes les causes potentielles seront déduites - cela pourrait être prouvé - mais il n'y aura pas de distinction entre des ensemble "et" et "ou" de causes.

Les ∝P et ∂P sont représentés par des attributs synthétisés. Par exemple :

∝(x Potentiel 1) ˜ ∂(x PotentialPossibleInitialCause 1)

En pratique, nous ne distinguons pas les causes de différents effets, bien que cela aurait été possible.

Nous ne donnons pas la grammaire pour la génération d'hypothèses, elle travaille d'une manière similaire à la contraposition.

Elimination d'hypothèse : Il n'y a pas de difficulté particulière ici parce que, pour parler simplement, cette transformation garde l'ordre de déduction des principes causaux.

Les grammaires ont été mises en œuvre sous la forme d'un ensemble de règles de production du premier ordre (environ 150 règles). A partir des 250 principes causaux, 400 principes inférentiels sont déduits. Seul un sous-ensemble des 250 principes initiaux sont relatifs à des quantités physiques, et méritent donc d'être transformé (les principes "fonctionnels" ne relèvent pas du raisonnement hypothétique).

3.2.4 Détail de la compilation de connaissance : des principes inférentiels au compilateur de SETR

La seconde étape part des principes inférentiels, et génère un ensemble de productions qui, lorsqu'appliquées à la description complète de la centrale, synthétise le système expert temps-réel (SETR). Cette transformation est relativement évidente. La seule astuce est de distinguer ce qui relève des informations temps-réel. Pour chaque principe inférentiel un ensemble de productions est généré, qui mentionnent dans leurs parties gauches la description structurelle, et dans leurs parties droites les schémas de règles 0^+.

Dix productions sont générées en moyenne pour chaque principe inférentiel. De 400 principes, on génère donc 4000 règles du premier ordre.

3.2.5 Détail de la compilation de connaissance : la génération du SETR

Le mécanisme est ici évident : les 4000 règles de production précédemment générées sont exécutées sur la description structurelle complète (environ 150000 "faits"). Cela produit environ 47000 règles dans la version actuelle.

3.2.6 La compilation de connaissances en pratique

Notre architecture a des avantages pratiques : il est possible de déconnecter la "maintenance conceptuelle", qui consiste à maintenir les principes et les connaissances génériques, de la "maintenance structurelle", qui consiste à maintenir la description structurelle.

Les étapes 1 et 2 sont exécutées purement hors-ligne. Cela signifie que ces étapes sont complètement indépendantes de la description structurelle de l'installation (les composants et leurs connexions). En général, le genre de composants dont une centrale est constituée ne change pas, ou peu fréquemment. Si un nouveau type de composant se présente, de nouveaux principes sont ajoutés, et les étapes 1 et 2 ré-exécutées. Ainsi, ces étapes peuvent être traitées dans notre laboratoire, et il est inutile d'installer les programmes correspondants sur le site de la centrale.

L'étape 3 dépend de la description structurelle. En pratique, les composants d'une centrale nucléaire change un peu périodiquement (une fois toutes les trois semaines en moyenne). Ainsi, la base de données contenant l'information structurelle doit être maintenue par les opérateurs lorsque des changement surviennent. A chaque fois que cela se produit, la troisième étape doit être ré-exécutée pour regénérer le système temps-réel. Nous n'avons pas mis en œuvre de mode de compilation incrémental, bien que cela aurait été possible (mais inutile). Cette étape prend environ 10 heures. Une fois terminée, le nouveau système temps-réel est lancé.

Le système temps-réel, y compris ses systèmes d'interface, fournit un diagnostic toutes les 10 secondes.

4 Autres aspects de KSE

Nous discutons ici diverses questions soulevées par notre architecture.

4.1 Performances

Comme dit plus haut, les modèles sont construits et les premières étapes de compilation effectuées dans notre laboratoire. Temps et mémoire consommés sont donc de peu d'importance.

La troisième étape requiert environ 10 heures, ce qui est compatible avec les "constantes de temps" de la conduite temps-réel. Nous savons de toute façon comment réduire ce temps par des techniques de compilation de règles, et aussi tout simplement en changeant d'ordinateur.

Le SETR effectue un diagnostic complet en 5 secondes (10 secondes avec les interfaces). Nous avons développé des techniques de compilation qui nous permettraient de réduire ce temps d'un facteur 15. Le lecteur ne doit pas être étonné de ces performances : des règles d'ordre 0^+ sont très simples. Ainsi, nous pouvons espérer utiliser des SETR plus volumineux, jusqu'à 500000 règles.

4.2 Justesse et complétude

Bien que nous ne l'ayons pas prouvé formellement, nous pensons que notre système donne toujours une réponse correcte, et fournit toujours à l'opérateur la meilleure vue possible sur l'état de l'installation, au moins en termes de causes initiales et de certaines propriétés fonctionnelles. Cependant, cela n'est vrai que dans un monde parfait. De manière évidente, la justesse d'une réponse repose fortement sur la justesse des modèles et des informations temps-réel fournies par le système d'information de la centrale. Nos expériences sur site durant deux années ont montré que le système peut fournir des réponses erronées principalement pour deux raisons : une information temps-réel est incorrecte, ou la description structurelle est incorrecte[1]. Les deux raisons sont intervenues avec des fréquences équivalentes.

Pour empêcher que de telles erreurs aient des conséquences, KSE met aussi en œuvre dans le SETR un composant qui vérifie les déductions en les confrontant aux informations disponibles. Ainsi, il peut trouver que quelque chose est incorrect, et informe l'opérateur dans ce cas. Il serait aussi possible de distinguer quelles déductions restent correctes malgré la contradiction identifiée. Cependant, trouver la cause d'une contradiction est un problème difficile. Comme mentionné ci-dessus, nous ne pouvons pas considérer le problème "pur" où la description structurelle serait parfaite. Ainsi, les principes du diagnostic basé sur les modèles doivent être réinterprétés ici.

Nous n'avons pas encore tout à fait résolu ce problème. Cependant, une étape a été accomplie grâce au système Melodia. Melodia est un vérificateur de règles 0^+ qui, à partir d'un ensemble de telles règles, produit toutes les clauses décrivant les mémoires de travail initiales conduisant à une contradiction. Dans notre cas, la mémoire de travail du SETR est constituée de l'information temps-réel. Nous avons obtenu des résultats très surprenants lorsque nous avons appliqué Melodia à quelques versions du SETR : des centaines de milliers de clauses d'incohérence étaient générées.

En y réfléchissant un peu, cela n'est pas étonnant. Les réponses du SETR sont correctes pourvu que les données soient correctes, mais rien n'assure dans sa conception qu'il en soit ainsi autrement. Ainsi, les clauses d'incohérence produites par Melodia sont des *contraintes sur les données physiques*. Si les clauses sont satisfaites par les données, alors les données peuvent être considérées comme correctes. Si une contradiction arrive malgré tout lors de l'utilisation du SETR, c'est qu'il y a très probablement une erreur dans la description structurelle à partir de laquelle le SETR a été généré. Au contraire, si les clauses sont violées, il y a un problème dans les données. Dans ce cas, des stratégies de diagnostic peuvent être utilisées pour isoler l'information fautive.

Ces idées n'ont pas encore été mises en œuvre. Cependant, nous pensons que cela est possible, malgré la taille de l'ensemble de clauses considéré. En particulier, vérifier les données par les 500000 clauses devrait être possible en quelques secondes; c'est un problème très similaire à la compilation de règles d'ordre 0.

5 Ce qui est important en IA "appliquée"

Nous pensons que ce travail peut fournit à d'autres utilisateurs de l'IA une technique de compilation de connaissances intéressante. Par delà cette constatation, nous aimerions faire quelques commentaires motivés par notre expérience.

Nous sommes partis dans ce travail des idées générales autour du paradigme "basé sur les modèles". Cependant, certains des principaux sujets de recherche dans ou autour de ce domaine

[1] Il arriva une fois qu'un principe causal soit incorrect. Cela a été immédiatement corrigé.

nous ont été d'une utilité presque nulle. Par exemple, nous n'avons pas utilisé de physique qualitative, ni de diagnostic basé sur les modèles.

Nous avons déjà mentionné que nous avons limité notre travail à la partie de la centrale où les processus physiques étaient simples, et où aucun raisonnement temporel n'était nécessaire[2]. Tout cela serait nécessaire pour étendre notre système à tous les composants (en particulier ceux impliquant des processus thermodynamiques). Mais nous avons déjà eu des problèmes critiques pour faire fonctionner notre système sur une échelle réaliste.

La même chose est vraie pour le diagnostic basé sur les modèles. Notre problème était très différent, puisque nous étions dans un monde où l'information sur les "composants fautifs" est disponible (en fait submergeante). De plus, la notion de composant "fautif" est différente, puisque dans notre cas cela ne signifie pas "cassé", mais ayant un comportement causant un effet non désiré du point de vue de la conduite globale. Néanmoins, nous avons mentionné au § 4 que nous devions quelquefois considérer un "composant cassé" : une mesure qui semble fausse.

Ainsi, il y a une sérieuse distorsion entre la "théorie" et la "pratique" : les problèmes réels sont très différents de ce que les problèmes-jouets suggèrent. Au-delà de ces différences, nous n'avons pas pu directement appliquer les travaux de recherche dans notre domaine pour deux raisons : la taille, et les erreurs.

La taille signifie que diverses techniques proposées dans la littérature de l'IA sont irréalistes : elles ont de trop mauvaises performances. Par exemple, nous n'avons pas utilisé un ATMS pour effectuer notre raisonnement hypothétique, justement parce que nous avions affaire à 12000 composants. C'est aussi la raison pour laquelle nous avons mis l'accent sur le développement de nouvelles techniques de compilation.

Les erreurs sont la conséquence de la taille. Par exemple, nous ne pouvons penser à aucune vérification statique *complète* de la description structurelle. Une erreur peut ne devenir visible que lorsque le système est utilisé sur l'installation réelle, mais pas avant. Or, dans tous les travaux sur le raisonnement basé sur les modèles, l'appareil sur lequel le système travaille est supposé être parfaitement décrit. Si çà n'est pas le cas, que se passe-t-il ? Que se passe-t-il, en particulier, lorsque l'on utilise des techniques sophistiquées ? Le système physique n'est pas le seul système susceptible de défaut lorsque l'on utilise un système de diagnostic, le système de diagnostic *aussi*.

Disons pour conclure ce paragraphe que nous n'entendons pas accuser les théoriciens de l'IA. Au contraire, nous pensons que la théorie est nécessaire, et il n'a pas été facile de développer nos propres techniques et notre cadre. Cependant, nous pensons que la théorie devrait se tourner, au moins pour ce qui concerne "l'IA appliquée", vers des questions plus réalistes, par exemple vers les conséquences de la taille et des erreurs.

Références

[Ancelin 87] J. Ancelin, J.P. Gaussot, P. Legaud. *Extra, a real-time knowledge-based alarm system*. ANS meeting "AI and other innovative computer applications in the nuclear industry", Snowbird 1987.

[2] Bien qu'une approche similaire à la notre soit possible dans ce cas, voir [Vinot 92].

[Bobrow 84] Artificial Intelligence, special issue on Qualitative Reasoning, Vol. 24, 1984.

[Charles 90] E. Charles. *Validation formelle d'une base de connaissances : le système MELODIA*. Thèse de l'Université Paris 6, 1990.

[Cherriaux 90] F. Cheriaux. *3SE, a real-time expert system for the electric sources monitoring in a nuclear plant*. Dixième conférence internationale "les systèmes experts et leurs applications", Avignon 1990.

[Console 90] L. Console, P. Torasso. *Integrating Models of the Correct Behavior into Abductive Diagnosis*. ECAI 90, Stockholm, Sweden.

[De Kleer 86] J. De Kleer. *An Assumtion-based Truth Maintenance System*. Artificial Intelligence Vol. 28, pp. 127-161, 1986.

[De Kleer 91] J. De Kleer & B. Williams. Artificial Intelligence, special issue on Qualitative Reasoning, Vol. 51, 1991.

[Dormoy 88a] J.L. Dormoy & O. Raiman. *Assembling a device*. AAAI 88, St-Paul Minneapolis. Aussi dans "Readings in Qualitative Reasoning about Physical Systems", De Kleer & Weld Eds.

[Dormoy 88b] J.L. Dormoy. *Controlling Qualitative Resolution*. AAAI 88, St-Paul Minneapolis.

[Raiman 89] O. Raiman. *Eléments d'une théorie du diagnostic*. Thèse de l'Université Paris 6, 1989.

[Reiter 87] R. Reiter. *A Theory of Diagnosis from First Principles*. Artificial Intelligence 32(1), pp. 57-96.

[Vinot 92] L. Vinot. *Gestion du temps dans un système expert temps réel de traitement d'alarmes.* Thèse de l'Université de Rennes, 1992.