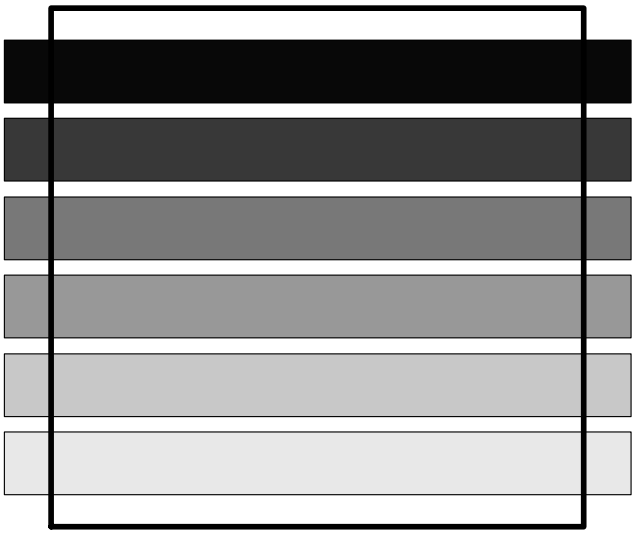
LES CAHIERS DU LANCI



COMPÉTENCE LOGIQUE ET CONNEXIONNISME

Nicolas Payette

UQÀM
Université du Québec à Montréal

No 2008-01

Le Laboratoire d'Analyse Cognitive de l'Information (LANCI) effectue des recherches sur le traitement cognitif de l'information. La recherche fondamentale porte sur les multiples conceptions de l'information.
Elle s'intéresse plus particulièrement aux modèles cognitifs de la classification et de la catégorisation, tant dans une perspective symbolique que connexionniste.
La recherche appliquée explore les technologies informatiques qui manipulent l'information. Le territoire privilégié est celui du texte.
La recherche est de nature interdisciplinaire. Elle en appelle à la philosophie, à l'informatique, à la linguistique et à la psychologie.

Volume 7, Numéro 2008-01 – Septembre 2008
Publication du Laboratoire d'Analyse Cognitive de l'Information
Directeurs : Luc Faucher, Jean-Guy Meunier, Serge Robert et Pierre Poirier
Université du Québec à Montréal
Document disponible en ligne à l'adresse suivante : www.lanci.uqam.ca
Tirage : 5 exemplaires

Dépôt légal – Bibliothèque Nationale du Canada Dépôt légal – Bibliothèque Nationale du Québec
ISBN-10 : 2-922916-20-0
ISBN-13 : 978-2-922916-20-1

COMPÉTENCE LOGIQUE ET CONNEXIONNISME

Nicolas Payette
Université du Québec à Montréal

Introduction

On peut généralement affirmer sans susciter trop de controverse que l'objectif premier de la logique est normatif. Ses règles d'inférence sont des règles de préservation de la vérité : elles visent à ce que celui qui part de prémisses vraies et ne fait que des inférences conformes à ces règles arrive à des conclusions vraies. Il est toutefois tentant de franchir le pas supplémentaire qui consiste à accorder aussi à la logique un statut descriptif : à affirmer que penser consiste à effectuer des opérations logiques sur des représentations symboliques. Il ne resterait plus qu'à décrire ces opérations et ces représentations pour que le tour soit joué et que nous sachions tout ce qu'il y a à savoir sur l'architecture de l'esprit. C'est une vieille idée (on la retrouve chez Occam, cf. Panaccio, 2004), mais elle a connu un second souffle avec l'avènement de l'ordinateur qui, lui, fonctionne effectivement de cette façon. Cette idée a servi de fer de lance à la révolution cognitiviste opérée au XX^{e} siècle. C'est probablement Jerry Fodor qui lui a donné sa formulation la plus influente, connue comme l'hypothèse du « langage de la pensée » (Fodor, 1975). Pour les besoins du présent texte, nous appellerons « classicisme » cette hypothèse et ses variantes. Le classicisme n'est ni sans problèmes (nous y reviendrons), ni sans compétition. L'alternative sur laquelle nous nous pencherons ici naît de l'idée que le fonctionnement du cerveau est pertinent pour comprendre la cognition. Elle a pour nom « connexionnisme » et pour stratégie de simuler ce fonctionnement en construisant des réseaux de neurones artificiels. C'est une idée plus récente que celle du langage de la pensée (elle a ses racines chez McCulloch et Pitts, 1943) et elle n'a commencé à exercer une influence significative que lors de la publication des deux volumes de *Parallel Distributed Processing* (McClelland, Rumelhart et le groupe de recherche PDP) en 1986. Sentant leur paradigme menacé, Fodor et le psychologue Zenon Pylyshyn (1988) ont tenté de montrer pourquoi le connexionnisme ne *pouvait pas* constituer une explication satisfaisante de nos capacités cognitives. Les connexionnistes furent piqués au vif et les réponses à Fodor, nombreuses (cf. Bechtel et Abrahamsen, 2002, ch. 6). Nous nous intéresserons ici à deux de ces réponses : celle de William Bechtel (1994; 1991; 2002) et celle de Jordan B. Pollack (1989; 1990).

Bechtel et Pollack ont tous deux décrit des architectures connexionnistes destinées à montrer comment Fodor et Pylyshyn ont sous-estimé les capacités des réseaux de neurones artificiels. Je tenterai pour ma part de montrer que les techniques développées par Pollack peuvent être utilisées pour améliorer les architectures conçues par Bechtel. Nous obtenons ainsi des réseaux plus puissants qui, je l'espère, ajouteront du poids du côté connexionniste de la balance.

Maintenant que nous avons fait connaissance avec les protagonistes, il est temps d'établir notre plan pour la suite des choses. Il nous faudra d'abord examiner un peu plus en détail le débat qui oppose Fodor et Pylyshyn aux connexionnistes. L'argument de Fodor et Pylyshyn repose sur l'incapacité supposée des réseaux connexionnistes à rendre compte de certaines propriétés de la pensée : notamment sa productivité et sa systématicité[1]. Nous aborderons brièvement ces propriétés. Nous examinerons ensuite la thèse de Bechtel, selon laquelle l'utilisation de symboles externes procure aux réseaux connexionnistes les capacités jugées nécessaires par Fodor et Pylyshyn. Pour illustrer sa théorie, Bechtel s'attaque à un type de capacités cognitives cher au classicisme : la compétence logique[2]. Nous examinerons l'architecture et les résultats obtenus par Bechtel pour deux tâches de logique[3]. La première consiste à identifier différentes formes d'arguments logiques et à juger de leur validité. La seconde est une tâche de déduction naturelle où le réseau, étant donné les prémisses et la conclusion d'un raisonnement, doit reconstruire les différentes étapes de ce raisonnement. Nous examinerons aussi les résultats obtenus en modifiant l'architecture de Bechtel pour utiliser des « représentations récursives distribuées », un type de représentations développé par Jordan B. Pollack (1989; 1990) pour répondre à Fodor. Nous nous pencherons ensuite sur la deuxième tâche à laquelle s'attaque Bechtel : la déduction naturelle. Ici encore, nous comparerons les résultats de Bechtel à ceux qui ont été obtenus en utilisant les représentations de Pollack.

Langage de la pensée vs. connexionnisme

Nous ouvrirons cette section en abordant une des failles que les connexionnistes croient avoir trouvé dans l'armure du classicisme : la question de la compétence logique. Bechtel décrit ainsi le point de départ du classicisme :

[1] Fodor et Pylyshyn parlent aussi de « cohérence inférentielle », mais celle-ci joue un rôle secondaire dans leur argument. Nous y reviendrons.
[2] Le tout premier programme d'intelligence artificielle symbolique était un « *Logic Theorist* » (Newell et Simon, 1956).
[3] Bechtel et Abrahamsen (1993, p. 184-185) présentent aussi une tâche consistant à compléter des enthymèmes. Nous ne l'aborderons pas ici.

> According to Fodor, cognitive performance requires an internal system of language-like representations and formal syntactic operations which can be applied to these representations. […] If the cognitive system has an overall language-like architecture, it makes sense to model deductive reasoning by specifying mental rules (comparable to the inference rules of natural deduction) that operate upon language-like mental representations. (Bechtel, 1994, p. 434)

En effet, si l'esprit fonctionne comme une machine logique, il semble permis de supposer que les humains peuvent faire intervenir ces opérations de base lorsque vient le temps d'exercer leur compétence logique. C'est un pas que Fodor et Pylyshyn n'hésitent pas à franchir en de nombreuses occasions. En voici une :

> It's a 'logical' principle that conjunctions entail their constituents (so the argument from P&Q to P and to Q is valid). Correspondingly, it's a psychological law that thoughts that P&Q tend to cause thoughts that P and thoughts that Q, all else being equal. (1988, p. 32)

Cela ne serait peut-être pas un problème pour le classicisme si ce n'était du fait que la compétence logique des humains laisse beaucoup à désirer, ce que montrent de nombreuses études empiriques (p. ex., Wason, 1966; Kern, Mirels et Hinshaw, 1983)[4]. Bechtel se fie aussi sur son expérience personnelle comme professeur de logique pour constater nos lacunes en ce domaine. La clause *ceteris paribus* ajoutées aux « lois psychologiques » comme celles que Fodor énonce ci-dessus est donc invoquée à outrance. Sommes-nous devant le cas khunien où l'accumulation d'anomalies justifie le remplacement d'une théorie ? Les connexionnistes le croient. Et ils croient aussi pouvoir expliquer de nombreuses capacités cognitives, y compris le raisonnement logique cher au classicisme, en les abordant comme des tâches de reconnaissance de formes (*pattern recognition*)[5].

Mettant à profit le dicton selon lequel la meilleure défense, c'est l'attaque, Fodor et Pylyshyn cherchent des raisons de principes pour lesquelles le connexionnisme ne pourra jamais remplacer l'approche classique de la cognition. Leur stratégie consiste à montrer que certaines caractéristiques observées de la pensée—sa productivité (PR), sa systématicité (SY) et sa cohérence inférentielle (CI)—impliquent les deux thèses suivantes, qui constituent l'essentiel des théories classiques (Fodor et Pylyshyn, 1988, p. 8) :

---

[4] À lui seul, ce fait remet en question le pouvoir explicatif du classicisme. Une étude de Kern et al. (1983) a montré, entre autres, que même des scientifiques ayant reçu une formation en logique ne réussissent le *modus tollens* qu'à 53%. Van Gelder et Niklasson (1994) montrent que le classicisme n'est pas en mesure de rendre compte de ces résultats de façon satisfaisante.

[5] Dans les mots de Bechtel : « Connectionists networks can naturally be interpreted as engaging in pattern recognition. [A]re connectionist networks capable of handling the broader range of tasks, including reasoning tasks, that we ordinarily think of as cognitive? If so, reasoning might be reconstructed as a kind of pattern recognition, and all of cognition might be brought under the same umbrella. » (1994, p. 435) Nous ne couvrirons pas ici ce « large éventail de tâches » ; seulement certains aspects du raisonnement logique.

(SC) Les représentations mentales ont une syntaxe et une sémantique combinatoires. Ce qui veut dire que : (a) les représentations « moléculaires » sont distinguées des représentations « atomiques »; (b) les représentations moléculaires ont des constituants syntaxiques qui sont eux-mêmes atomiques ou moléculaires; et (c) le contenu sémantique d'une représentation moléculaire est fonction de sa structure et du contenu sémantique de ses constituants.

(SS) Les processus mentaux sont sensibles à la structure combinatoire—telle que décrite en (SC)—des représentations sur lesquelles ils opèrent.

Le raisonnement de Fodor et Pylyshyn pour rejeter le connexionnisme (CX) s'établit donc ainsi :

(1)	PR ⊃ (SC & SS)	} *Prémisses.*
(2)	SY ⊃ (SC & SS)	
(3)	PR	} *Par observation.*
(4)	SY	
(5)	(SC & SS)	} *Par modus ponens :* (1),(3) *ou* (2),(4).
(6)	(SC & SS) | CX	} *Par définition.*
(7)	~CX	} *Par syllogisme alternatif :* (5), (6).

La stratégie pour un connexionniste consisterait plutôt à montrer que (CX & PR & SY), ce qui est incompatible avec (7), mais aussi avec les implications (1) et (2), puisqu'on aurait alors à la fois la négation de leurs conséquents (~(SC & SS), par CX et (6)) et l'affirmation de leurs antécédents (PR et SY)[6]. C'est ce que nous allons essayer de faire, mais il nous faut d'abord nous pencher sur les propriétés en question.

La productivité

Un système cognitif possède la propriété de productivité si sa structure est telle que, *en principe*, il peut comprendre et générer un nombre illimité de représentations. Il n'y a bien sûr aucun système qui puisse *en fait* accomplir cet exploit. La notion repose sur une conditionnelle contre-factuelle : *si* les ressources du système (mémoire, temps, motivation, etc.) étaient illimitées,

[6] Une autre stratégie serait de nier (3) et (4). Rumelhart et McLelland par exemple, comme le notent Fodor et Pylyshyn (1988, p. 23), nient (3) : « [We] do not agree that [productive] capabilities are of the essence of human computation. As anyone who has ever attempted to process sentences like 'The man the boy the girl hit kissed moved' can attest, our ability to process even moderate degrees of center-embedded structure is grossly impaired relative to an ATN [Augmented Transition Network] parser…. What is needed, then, is not a mechanism for flawless and effortless processing of embedded constructions… The challenge is to explain how those processes that others have chosen to explain in terms of recursive mechanisms can be better explained by the kinds of processes natural for PDP networks. » (1986, p. 199). Christiansen (1992), de même que Waskan et Bechtel (1997), par exemple, contestent aussi l'ubiquité de la productivité et de la systématicité dans la cognition.

le système pourrait générer/comprendre un nombre illimité de représentations. Le moins que l'on puisse dire est que ce genre d'idéalisations est controversé. Certains soutiennent même que les énoncés conditionnels contraires aux faits n'ont pas de valeur de vérité (par exemple, Dummett, 1982). La linguistique générativiste repose néanmoins sur l'idée qu'il en va ainsi de nos capacités linguistiques et elle a tenté d'apporter un certain appui empirique à cette thèse (Chomsky, 1968). Fodor et Pylyshyn soutiennent que c'est aussi le cas pour nos capacités cognitives (i.e. que le « langage de la pensée » est productif) et que le classicisme explique pourquoi il en est ainsi. En effet, si l'esprit fonctionne comme un ordinateur (i.e. comme une machine de Turing), il suffit d'ajouter de la mémoire et le tour est joué; nul besoin de modifier la « structure computationnelle » de la machine. Pour les connexionnistes, par contre, ça n'est pas aussi simple : ajouter de la mémoire implique d'ajouter des neurones (des *unités*) et donc de nouvelles connexions dans le réseau, ce qui modifie sa structure. En conséquence, selon Fodor et Pylyshyn :

> Connectionist cognitive architectures cannot, by their very nature, support an expandable memory, so they cannot support productive cognitive capacities. The long and short is that if productivity arguments are sound, then they show that the architecture of the mind can't be Connectionist. (1988, p. 23)

Les connexionnistes admettent qu'il est impossible d'ajouter de la mémoire à leurs architectures sans modifier ces dernières. Ce que certains d'entre eux n'admettent pas, toutefois, c'est que cela les prive nécessairement de capacités productives.

La systématicité

La notion de productivité, puisqu'elle repose sur une idéalisation dont la légitimité ne fait pas l'unanimité, n'offre au classicisme qu'un soutient partiel. Fodor et Pylyshyn assurent leurs arrières en faisant appel à la notion de « systématicité », élaborée par eux pour les besoins de la cause (cf. Niklasson et Van Gelder, 1994), espérant ainsi faire échec au connexionnisme tout en reposant sur des bases moins controversées.

Fodor et Pylyshyn utilisent la même stratégie pour établir la systématicité que pour établir la productivité. Ils partent de celle du langage pour arriver à celle de la pensée. Ils présentent ainsi la systématicité linguistique :

> What we mean when we say that linguistic capacities are *systematic* is that the ability to produce/understand some sentences is *intrinsically* connected to the ability to produce/understand certain others. (1988, p. 25, italiques originales)

C'est-à-dire que la capacité à produire/comprendre la phrase « Jean aime la fille » est intrinsèquement liée à la capacité à produire/comprendre la phrase « La fille aime Jean ». Il suffit de maîtriser une structure linguistique pour maîtriser toutes les phrases ayant la même structure (si on en connaît les constituants).

Ce qui vaut pour le langage vaudrait aussi pour la pensée. On peut comprendre la phrase « Jean aime la fille » seulement si on est capable de penser que Jean aime la fille. Et si on est capable de penser que Jean aime la fille, on sera aussi capable de penser que la fille aime Jean. Encore ici, c'est la correspondance structurelle entre les deux pensées qui permet la systématicité. Mais qu'est-ce qui assure cette relation structurelle? Et que doit-on en conclure? Selon Fodor et Pylyshyn :

> [T]he two mental representations, like the two sentences, *must be made of the same parts*. But if this explanation is right (and there don't seem to be any others on offer), then mental representations have internal structure and there is a language of thought. So the architecture of the mind is not a Connectionist network. (1988, p. 26, italiques originales)

Rappelons-nous ici de la thèse classiciste (SC), selon laquelle les représentations mentales ont une syntaxe et une sémantique combinatoires. Ce que Fodor et Pylyshyn nous disent, c'est que cette compositionalité (absente dans le connexionnisme) est nécessaire à la systématicité[7]. Nous verrons toutefois qu'il est possible pour un réseau connexionniste de faire preuve de systématicité sans faire appel au genre de compositionalité qu'ont en tête Fodor et Pylyshyn.

Tim van Gelder (1990) distingue la compositionalité *syntaxique* de la compositionalité *fonctionnelle*. La compositionalité syntaxique exige que les constituants d'une représentation complexe soient instanciés dans celle-ci. C'est celle du classicisme : « We […] stipulate that for a pair of expression types E1, E2, the first is a *Classical* constituent of the second *only if* the first is tokened whenever the second is tokened. » (Fodor et McLaughlin, 1990, p. 186, italiques originales) La compositionalité fonctionnelle, elle, exige simplement que les constituants d'une expression complexe puissent être récupérés à partir de celle-ci. Nous verrons que c'est exactement le genre de compositionalité dont font preuve les représentations développées avec les réseaux RAAM (*recursive auto-associative memory*) de Pollack et que ce genre de compositionalité est suffisant pour implémenter la systématicité.

Quelques mots sur la cohérence inférentielle

[7] Fodor et Pylyshyn vont jusqu'à dire que la compositionalité et la systématicité sont deux aspecta d'un même phénomène (p. 28).

Avant d'aller plus loin, il nous faut aborder brièvement la question de la cohérence inférentielle, une autre propriété que Fodor et Pylyshyn croient essentielle à la cognition. Ils nous la décrivent ainsi :

> [I]nferences that are of similar logical type ought, pretty generally, to elicit correspondingly similar cognitive capacities. You shouldn't, for example, find a kind of mental life in which you get inferences from P&Q&R to P but you don't get inferences from P&Q to P. (1988, p. 32)

Fodor et Pylyshyn ne nient pas qu'il puisse y avoir des réseaux de neurones artificiels qui fassent preuve de cohérence inférentielle. Ce qui les trouble, c'est qu'il *peut* aussi y en avoir qui ne fassent *pas* preuve de cohérence inférentielle. Le connexionnisme permet d'imaginer des esprits qui sont prêts à inférer *Jean est allé au magasin* de *Jean et Marie sont allés au magasin*, mais pas de *Jean et Marie et Susanne* sont allés au magasin. Ce n'est pas le cas avec les modèles classiques :

> Given a notion of logical syntax—the very notion that the Classical theory of mentation requires to get its account of mental processes off the ground—it is a *truism* that you don't get such minds. Lacking a notion of logical syntax, it is a *mystery* that you don't. (1988, p. 33)

Il suffit toutefois d'adopter une perspective évolutionniste pour que le « mystère » évoqué par Fodor et Pylyshyn s'éclaircisse un peu. Si on remonte assez loin dans la phylogénie, on trouvera éventuellement des ancêtres qui ne faisaient pas preuve de cohérence inférentielle. Ces ancêtres avaient néanmoins certaines capacités cognitives, si limitées fussent-elles, implémentées dans un système nerveux primitif. Notre cerveau actuel et ses capacités logiques ont évolué graduellement à partir de celles de ces ancêtres. Quand on prend un peu de recul, une architecture générale qui puisse potentiellement s'appliquer à toute la phylogénie—puisqu'elle ne suppose pas *a priori* des capacités comme la cohérence inférentielle—apparaît comme un avantage et non comme un inconvénient. Dans cette perspective, c'est plutôt le classicisme qui génère un mystère : quand donc apparaît cet extraordinaire « langage de la pensée », qui n'admet d'entrée de jeu aucune faille dans la cognition[8]? Le portait que peignent Fodor et Pylyshyn semble laisser bien peu de place à un développement graduel de la cognition. Nous ne reviendrons pas sur cette question.

[8] Fodor et Pylyshyn justifient leur position en affirmant que : « Infraverbal cognitive architecture mustn't be so represented as to make the eventual acquisition of language in philogeny and in ontogeny require a miracle. » (1988, p. 27) Ce faisant, ils ne font toutefois que repousser le « miracle » plus loin dans la phylogénie. S'il faut le « langage de la pensée » pour expliquer l'apparition du langage, que faut-il pour expliquer celle du langage de la pensée ?

Les réseaux de Bechtel

En quelques mots, l'idée de départ de Bechtel est la suivante :

> [T]he systematicity of human thought might better be explained as resulting from the fact that we have learned natural languages which are themselves syntactically structured. According to this view, symbols of natural language are external to the cognitive processing system and what the cognitive system must learn to do is produce and comprehend such symbols. (1994, p. 433)

Une fois le problème formulé ainsi, la tâche ne consiste plus à expliquer la systématicité (ou la productivité) cognitive d'un esprit abstrait, désincarné, mais plutôt celle d'un système composé de l'individu et de l'environnement (matériel et culturel) dans lequel il est situé[9]. C'est une idée que Bechtel a exploitée ailleurs (1996a; 1996b) et qui se situe dans la mouvance de l'externalisme actif (cf. Clark, 1997).

Pour parvenir à exploiter cette systématicité externe, un réseau de neurones artificiels doit s'appuyer sur ses capacités de reconnaissance de formes. On trouve déjà cette idée dans *PDP* (McClelland *et al.*, 1986, p. 45), qui donnent comme exemple l'opération de multiplication. Peu d'entre nous sommes capables de réussir la multiplication de deux nombres supérieurs à 100 sans faire appel à du papier, un crayon, et un système de représentations externes (par exemple, les chiffres arabes). Une fois ces éléments en place, on enclenche une boucle de rétroaction : on reconnaît la forme du problème et on pose une action en conséquence, qui en modifie la forme. Puis on recommence jusqu'à ce que le problème soit résolu. C'est ce qui se passe dans le cas de la multiplication : on reconnaît la forme du problème de multiplication et on voit qu'il faut d'abord multiplier ensemble les chiffres des unités des deux nombres et poser la réponse sous la ligne, ce qui transforme le problème et nous permet de voir quelle doit être l'opération suivante.

Cette capacité à reconnaître la forme d'un problème n'est pas limitée aux problèmes arithmétiques : on peut aussi voir un argument logique et reconnaître qu'il s'agit d'un *modus ponens* et que c'est un argument valide. C'est cette capacité qu'explore Bechtel avec son premier réseau[10].

---

[9] Cette réponse n'est pas sans rappeler le « *system reply* » évoqué par Searle en ce qui concerne le problème de la chambre chinoise (Searle, 1980). C'est aussi un genre de réponse qui est susceptible d'être rejetée par certain néo-wittgensteiniens (par exemple, Bennett et Hacker, 2003) sous prétexte que les prédicats psychologiques sont dénués de sens lorsque appliqués à des systèmes sub- ou supra-personnels. Voir (Poirier et Payette, 2007) pour une contre-contre-objection.

[10] Ce réseau est décrit par Bechtel à trois endroits différents : (Bechtel et Abrahamsen, 1991; Bechtel, 1994; Bechtel et Abrahamsen, 2002). La description la plus complète se trouve dans (1991) mais certains détails ne sont abordés que dans (2002).

Premier réseau : reconnaissance et évaluation d'arguments

Bechtel construit son ensemble de problèmes à partir quatre formes d'arguments : modus ponens (MP), modus tollens (MT), syllogisme alternatif (SA) et syllogisme disjonctif (SD). Il y a une version valide et une version invalide pour chaque forme d'arguments. Celles-ci sont présentées dans le tableau suivant :

Tableau 1 – Schémas d'arguments logiques

	Modus ponens (MP)	Modus tollens (MT)	Syllogisme alternatif (SA)		Syllogisme disjonctif (SD)	
Formes valides	$p \supset q$ p ——— $\therefore q$	$p \supset q$ $\sim q$ ——— $\therefore \sim p$	$p \vee q$ $\sim p$ ——— $\therefore q$	$p \vee q$ $\sim q$ ——— $\therefore p$	$p \mid q$ p ——— $\therefore \sim q$	$p \mid q$ q ——— $\therefore \sim p$
Formes invalides	$p \supset q$ q ——— $\therefore p$	$p \supset q$ $\sim p$ ——— $\therefore \sim q$	$p \vee q$ p ——— $\therefore \sim q$	$p \vee q$ q ——— $\therefore \sim p$	$p \mid q$ $\sim p$ ——— $\therefore q$	$p \mid q$ $\sim p$ ——— $\therefore p$

Dans l'ensemble de problèmes, les variables *p* et *q* des schémas ci-dessus sont remplacés par des constantes logiques : A, B, C et D, et leurs négations, ~A, ~B, ~C et ~D. On obtient donc 576 arguments possibles : (12 schémas) × (8 possibilités pour *p*) × (6 possibilités pour *q*). Il y a moins de possibilités pour *q* parce que la constante utilisée pour *p* et la négation de celle-ci sont exclues.

L'ensemble ensuite divisé en trois sous-ensembles de 192 arguments (E_1..E_3) qui serviront à l'entraînement et au test des capacités du réseau[11].

Architecture

Bechtel utilise un réseau *feedforward* à quatre couches, entraîné par rétropropagation à l'aide du programme **bp** exposé au chapitre 5 du *Handbook* PDP (McClelland et Rumelhart, 1988)[12]. La couche d'entrée reçoit une représentation de l'argument, encodée sur 14 unités. Les deux couches intermédiaires (dites « cachées ») sont composées de 10 unités. La couche de sortie, elle, doit représenter la forme et la validité de l'argument sur 3 unités.

[11] Bechtel s'assure que chacun des sous-ensembles contient au moins un argument pour chaque schéma, valide et invalide. Cette contrainte facilite la tâche du réseau et je ne l'appliquerai pas lorsque viendra le temps de la reproduire.
[12] Le lecteur peu familier avec les réseaux de neurones artificiels est invité à consulter la seconde édition du livre de Bechtel et Abrahamsen (2002), qui constitue une excellente introduction au domaine. Les détails techniques du genre d'architecture utilisé par Bechtel peuvent aussi être retrouvés dans le *Handbook* et le premier volume de *PDP* (McClelland et Rumelhart, 1988; Rumelhart *et al.*, 1986).

Les représentations sur la couche d'entrée sont encodées sous forme binaire et utilisent les composantes suivantes :

Tableau 2 - Encodage binaire des composantes pour le premier réseau de Bechtel

Entrées					
Opérateurs		Constantes			
⊃	11	A	001	~A	101
v	01	B	010	~B	110
|	10	C	011	~C	111
		D	000	~D	100

Sorties			
Formes		Valide ?	
MP	01	Oui	1
MT	10	Non	0
SA	11		
SD	00		

Pour représenter un argument complet sur la couche d'entrée du réseau, des valeurs d'activation de 0 ou 1—correspondant aux différentes composantes de l'argument—sont fournies aux unités de la couche d'entrée de façon à former une représentation binaire. Par exemple, le *modus ponens* invalide « Si A alors non-C, or non-C, donc A » est représenté comme suit :

Tableau 3 - Représentation d'un *modus ponens* invalide sur la couche d'entrée

Prémisse 1								Prémisse 2			Conclusion		
A			⊃		~C			~C			A		
0	0	1	1	1	1	1	1	1	1	1	0	0	1

Ces valeurs d'activation sont ensuite propagées le long des connexions du réseau et modulées par les unités des couches intermédiaires. Le réseau devrait alors « répondre », sur sa couche de sortie, qu'il s'agit d'un *modus ponens* invalide :

Tableau 4 – Représentation d'un *modus ponens* invalide sur la couche de sortie

Forme		Valide?
MP		Non
0	1	0

Notons que, vu que la valeur d'activation des unités de la couche de sortie peut théoriquement être égale à n'importe quel nombre réel entre 0 et 1, Bechtel utilise un seuil de 0,5 : toute valeur d'activation plus petite que 0,5 est ramenée à 0, toute autre valeur à 1.

Résultats

Bechtel accorde d'abord à son réseau une première période d'entraînement sur l'ensemble E_1 (le premier tiers de l'ensemble de problèmes, composé de 192 arguments). Après 3000 époques[13], le réseau maîtrise parfaitement l'ensemble E_1 : il réussit à identifier correctement, sur sa couche de sortie, la forme et la validité de chacun des arguments qui lui sont présentés sur sa couche d'entrée. Pour vérifier sa capacité à généraliser, le réseau est ensuite testé sur l'ensemble E_2, où il répond correctement à 76% des problèmes[14]. Le réseau est ensuite entraîné pendant 5000 époques sur la réunion de E_1 et E_2, après quoi il répond correctement à 98,9% des problèmes (380/384). Finalement, le réseau est testé sur les problèmes de E_3, qu'il n'a encore jamais rencontrés, où sa performance est de 83,9% (161/192). Le tableau suivant résume ces résultats[15] :

Tableau 5 – Résultats de Bechtel pour la reconnaissance et l'évaluation d'arguments

Entraînement E_1	3000 époques	
Test E_1	192 / 192	100%
Test E_2	146 / 192	76%
Entraînement E_1+E_2	5000 époques	
Test E_1+E_2	380 / 384	98,9%
Test E_3	161 / 192	83,9%

C'est une performance fort respectable, mais nous verrons plus loin qu'elle peut être améliorée grâce à l'utilisation des représentations RAAM. Pour l'instant, nous nous attarderons au deuxième réseau de Bechtel.

Deuxième réseau : déduction naturelle

[13] Lors d'une « époque », tous les éléments de l'ensemble d'entraînement sont présentés séquentiellement au réseau. Lors de chaque présentation, l'algorithme d'apprentissage par rétropropagation est appliqué aux poids des connexions du réseau pour les ajuster de façon à diminuer l'erreur, c'est-à-dire la différence entre la sortie attendue et la sortie effective. Bechtel et Abrahamsen (2002, §3.2.2) offrent une excellente introduction à l'apprentissage par rétro-propagation.

[14] Bechtel nous dit (Bechtel, 1994; Bechtel et Abrahamsen, 1991, 1993, 2002) que son réseau répond correctement à 139 problèmes sur 192, soit 76%, ce qui est impossible puisque 139/192=72%; c'est plutôt 146 sur 192 qui donne 76%. Nous appliquerons ici le principe de charité et prendrons la meilleure interprétation possible, soit 146/192=76%.

[15] Berkeley et ses collègues ont reproduit la même tâche avec une architecture légèrement différente, qui leur a permis d'offrir une analyse détaillée de la propagation de l'activation dans les unités cachées du réseau. Voir (Berkeley *et al.*, 1995; Dawson, Medler et Berkeley, 1997) et aussi (Bechtel et Abrahamsen, 2002, p. 112-113) pour une courte discussion.

La deuxième tâche entreprise par Bechtel[16] repose aussi sur l'utilisation de symboles externes. Il s'agit cette fois d'entraîner un réseau à effectuer des dérivations logiques : à partir de prémisses données, trouver les étapes nécessaires pour arriver à une conclusion donnée.

Ici aussi, Bechtel s'appuie sur son expérience de professeur de logique. Il remarque que, lorsqu'il enseigne à effectuer des dérivations, il est souvent appelé, pour des raisons pédagogiques, à verbaliser ses stratégies de dérivations—à les énoncer sous formes de règles. Il admet toutefois que :

> [W]hen I engage in offering such explanations, I strongly suspect that I am confabulating. Having done innumerable derivations, when I first look at natural deduction problems, especially relatively simple ones, I immediately *see* what to do. (1994, p. 445, italiques originales)

Bref, son impression est qu'il procède surtout par reconnaissance de formes et que les règles de dérivation transmises à ses étudiants sont reconstruites *a posteriori*. Cela semble donc être une tâche appropriée pour un réseau de neurones, surtout qu'il est possible d'exploiter le genre de boucles de rétroactions évoquées plus haut puisque chaque ligne ajoutée à dérivation—chaque nouvelle inférence—change la forme du problème à traiter.

Le tableau suivant présente les schémas de dérivation (I-XX) utilisés par Bechtel pour générer un ensemble de problèmes. Les premières lignes contiennent les prémisses données au réseau ; les lignes grisées contiennent les inférences à effectuer et la dernière ligne contient la conclusion à laquelle il doit parvenir.

[16] Celle-ci est décrite en détail dans (Bechtel, 1994) et résumée dans (Bechtel et Abrahamsen, 2002)

Tableau 6 - Schémas de dérivation

I	II	III	IV	V	VI	VII	VIII	IX	X
$p \vee q$	$p \vee q$	$p \supset q$	$p \supset q$	$p \vee q$	$p \supset q$	$p \,\&\, q$	$p \,\&\, q$	$\sim p \,\&\, q$	$\sim p \,\&\, q$
$q \supset r$	$q \supset r$	$q \supset r$	$q \supset r$	$q \supset r$	$q \supset r$	$p \supset \sim r$	$p \supset r$	$p \vee r$	$p \vee \sim r$
$\sim p$	$\sim p$	p	p	$\sim p \,\&\, s$	$p \,\&\, q$	$r \vee s$	$q \supset s$	$r \supset s$	$r \vee \sim s$
q	***q***	***q***	***p ⊃ q***	***p***	***p***	***p***	***p***	***~p***	***~p***
r	***r***	***r***	***q ⊃ r***	***q***	***q***	***~r***	***r***	***r***	***~r***
$r \vee s$	$r \,\&\, q$	$r \vee s$	***p***	r	r	***s***	***q***	***s***	***s***
			q			***q***	***s***	***q***	***q***
			r			$s \,\&\, q$	$r \,\&\, s$	$s \,\&\, q$	$s \,\&\, q$
			$r \,\&\, q$						

XI	XII	XIII	XIV	XV	XVI	XVII	XVIII	XIX	XX
$p \supset q$	$p \vee \sim q$	$p \vee q$	$p \supset \sim q$	$p \vee \sim q$	$p \supset \sim q$	$p \supset \sim q$	$p \,\&\, q$	$p \vee \sim q$	$p \supset q$
$p \supset r$	$p \vee \sim r$	$q \supset r$	$q \vee \sim r$	$q \vee r$	$q \vee r$	$q \vee r$	$p \supset r$	$q \vee r$	$q \supset \sim r$
$r \supset s$	$r \vee s$	$r \supset s$	$r \vee s$	$\sim p$	p	$p \,\&\, q$	$r \supset s$	$r \supset s$	$r \vee s$
p	$\sim p$	$\sim p$	p	***~q***	***~q***	***p***	***p***	***~p***	p
q	***~q***	***q***	***~q***	***r***	***r***	***~q***	***r***	***q***	***q***
r	***~r***	***r***	***~r***	$r \vee s$	$r \,\&\, \sim q$	r	***s***	***r***	***~r***
s	***s***	s	s				***q***	***s***	s
$s \,\&\, q$	$s \,\&\, \sim r$						$s \,\&\, q$	$s \,\&\, \sim q$	

Les variables (*p*, *q*, *r* et *s*) utilisées dans les schémas sont remplacées par les constantes A, B, C, D pour générer un ensemble de problèmes. Dans un premier temps, Bechtel n'utilise pas de négations pour ses constantes; elles seront introduites plus loin. Voici un exemple de dérivation générée avec le schéma I :

Tableau 7 - Exemple de dérivation

Schéma I			Dérivation	Opération
Prémisse	1.	$p \vee q$	A ∨ B	
Prémisse	2.	$q \supset r$	B ⊃ C	
Prémisse	3.	$\sim p$	~A	
Inférence	4.	q	B	:1,3 ∨-elim
Inférence	5.	r	C	:2,4 ⊃-elim
Conclusion	6.	$r \vee s$	C ∨ D	:5 ∨-intro

Un problème comme celui-ci sera traité en deux étapes par le réseau (une étape pour chaque inférence). Dans un premier temps, les prémisses et la conclusion sont présentées sur la couche d'entrée du réseau, qui doit fournir une représentation de la proposition requise à l'étape 1 sur sa couche de sortie (ici, « B »). Dans un deuxième temps, on fournit au réseau les prémisses, la

conclusion ET l'étape 1 déjà complétée (on suppose que le réseau a « écrit » la proposition sur sa « feuille »[17]). Le réseau doit alors fournir une représentation de l'étape 2 sur sa couche de sortie.

Architecture

Toutes les propositions d'une dérivation sont représentées de la même façon (à l'aide de 13 unités) qu'il s'agisse d'une prémisse, de la conclusion ou d'une inférence (en entrée comme en sortie). Ici encore, Bechtel utilise des représentations binaires pour chaque composante d'une proposition :

Tableau 8 - Encodage binaire des composantes pour le deuxième réseau de Bechtel

Opérateurs		Constantes			
&	100	A	00001	~A	10001
⊃	010	B	00010	~B	10010
v	001	C	00100	~C	10100
		D	01000	~D	11000

Une proposition moléculaire, par exemple ~A & B, est représentée de la façon suivante :

Tableau 9 - Encodage binaire de ~A & B

Première composante					Opérateur			Deuxième composante				
~A					&			B				
1	0	0	0	1	1	0	0	0	0	0	1	0

Pour une proposition atomique, par exemple ~C, on laissera simplement des zéros dans les deux premières sections :

Tableau 10 - Encodage binaire de ~C

Première composante					Opérateur			Deuxième composante				
NIL[18]					NIL			~C				
0	0	0	0	0	0	0	0	1	0	1	0	0

La couche d'entrée du réseau est composée de 8 blocs de 13 unités (pour un total de 104 unités) : un bloc pour chaque ligne de la dérivation (incluant prémisses et conclusion). Lorsqu'un problème est présenté au réseau sur sa couche d'entrée, les blocs correspondant aux lignes

[17] Le programme d'entraînement fournit toujours au réseau les propositions correctes pour les étapes complétées.
[18] Ici comme plus loin, le symbole « NIL » signifie : néant, nul, rien, non applicable. Sur la couche d'entrée d'un réseau, il se traduira par une série de zéros.

inutilisées (soit parce qu'il s'agit d'une étape qui reste à faire, soit parce qu'il s'agit d'une dérivation de moins de 8 lignes) sont remplis avec des zéros. Le réseau comporte une couche intermédiaire de 20 unités et une couche de sortie constituée d'un bloc de 13 unités, sur laquelle il doit représenter la proposition correspondant à l'étape suivante de la dérivation.

Résultats

Pour une première expérience, Bechtel n'utilise que les schémas I–XIV. Les dérivations sont générées en remplaçant les variables par des constantes, mais aussi en permutant les trois premières prémisses pour chaque schéma : 1, 2, 3 ; ou bien 2, 1, 3 ; ou bien 3, 1, 2. Ces permutations sont effectuées pour éviter que le réseau puisse se fier sur l'ordre des prémisses pour réussir ses dérivations. On arrive ainsi à 24 dérivations possibles pour chaque schéma, ce qui fait un total de 1008 dérivations. Celles-ci sont séparées en deux ensembles : E_1 et E_2. Les trois-quarts des dérivations iront dans E_1 et le quart restant dans E_2, qui servira à tester la généralisation. On a un total de 1800 inférences (des « étapes de dérivation ») à faire dans E_1 et 936 inférences dans E_2. Pour 500 époques d'entraînement sur E_1, les résultats de Bechtel sont les suivants[19] :

Tableau 11 - Résultats de Bechtel pour la première expérience de déduction naturelle

Entraînement E_1	500 époques	
Test E_1	1800 / 1800	100%
Test E_2	767 / 936	81,9% (90,5% WTA[20])

Non seulement le réseau réussit-il à maîtriser parfaitement E_1, mais il est capable de généraliser assez bien à E_2. Rappelons que les dérivations dans E_2 utilisent les mêmes schémas que celles de E_1, mais avec des constantes et des permutations de prémisses différentes.

Pour une deuxième expérience, Bechtel introduit quelques négations. De nouvelles dérivations pour lesquelles soit *q*, soit *p* et *r*, soit *q*, *r* et *s* sont niées, sont ajoutées aux ensembles de problèmes[21]. Cette fois-ci, le réseau utilisé possède une couche intermédiaire composée de 40

[19] Voir (Bechtel, 1994, p. 452) pour une description plus détaillée de ces résultats, particulièrement en ce qui concerne les types d'erreurs commises par le réseau sur E_2.

[20] C'est le pourcentage que Bechtel aurait obtenu s'il avait employé la méthode « *winner take all* », où les connexions entre les unités d'un même groupe sont organisées de façon à ce qu'une seule des unités de ce groupe soit activée à la fois.

[21] Bechtel applique la contrainte suivante à la constitution de son ensemble de test : « if a substitution set in which one assignment of sentential constants to sentential letters was chosen for the test set, the other three substitution sets differing in assignments of negations were also chosen. Thus, if the substitution set in which A, D, C, B were chosen to replace p, q, r, s was elected, then so were the substitution sets in which A, ~D, C, B; ~A, D, ~C, B; and A, ~D,

unités plutôt que 20. Il lui suffit de 100 époques pour maîtriser cette version étendue de E_1. La généralisation à E_2 se fait à hauteur de 84,6%. Cette fois-ci, Bechtel ajoute un troisième ensemble de problèmes, E_3, créé à partir des schémas XV–XX, jusqu'alors inutilisés. De nouvelles dérivations sont générées en utilisant uniquement les combinaisons de constantes B, A, D, C; A, ~D, B, C; ~D, B, ~A, C; et B, ~D, ~A, ~C pour remplacer *p*, *q*, *r*, *s*, ce qui donne 252 nouvelles inférences. Le réseau effectue correctement 198 d'entre elles, ce qui est impressionnant vu qu'il s'agit de dérivations générées à partir de *nouveaux* schémas *sur lesquels le réseau n'a jamais été entraîné.*

Tableau 12 - Résultats de Bechtel pour la deuxième expérience de déduction naturelle

Entraînement E_1	100 époques	
Test E_1	7 786 / 7 786[22]	100%
Test E_2	2 112 / 2 496	84,6% (93,6% WTA)
Test E_3	198 / 252	78,6%

Le moment est venu de voir ce qui peut être fait en appliquant le concept de représentations récursives distribuées à ces problèmes.

Les réseaux RAAM

Si la compositionalité syntaxique de nos représentations mentales est objet de controverse, celle de nos représentations linguistiques ne fait toutefois pas de doute : les phrases de notre langage et les propositions de notre logique ont une syntaxe compositionnelle. Les réseaux de Bechtel ont montré qu'il était possible, jusqu'à un certain point, d'exploiter cette compositionalité externe. Ils montrent toutefois leurs limites lorsqu'on envisage de traiter des propositions plus complexes. En effet, considérons le nombre d'unités qui seraient nécessaires au premier réseau de Bechtel pour représenter sur la couche d'entrée le *modus ponens* suivant :

(A∨~B) ⊃ ~(C|D), or (A∨~B), donc ~(C∨D)

Si on utilisait la même méthode d'encodage que Bechtel, il faudrait 39 unités. Ce n'est pas un nombre d'unités exagéré, mais un problème survient si on veut que le réseau apprenne à reconnaître qu'il s'agit là de la même forme d'argument qu'un *modus ponens* plus simple, que l'on

~C, ~B served as the replacements. » (p. 453) Une autre variante de l'expérience est décrite dans (Bechtel, 1994) mais nous ne l'aborderons pas ici.

[22] Bechtel ne nous fournit pas ce chiffre. Je l'ai estimé au *pro rata* du nombre d'inférences (2496 dans E_2) par rapport au nombre de dérivations (268 dans E_1 et 836 dans E_2).

suppose encore encodé sur 13 unités. Il y a bien sûr moyen d'organiser les représentations de façons à faire ressortir la communauté de forme entre les deux arguments, mais cela ne fait que repousser plus loin le problème, sans vraiment le régler. Si on veut traiter des expressions encore plus complexes—ou simplement organisées différemment—tout est à refaire. C'est ce genre de problème que Fodor et Pylyshyn pointent du doigt lorsqu'ils disent que les architectures connexionnistes ne peuvent bénéficier d'un ajout de ressources sans modification de leur structure. C'est également ce genre de problème que Pollack cherchait à résoudre lorsqu'il a conçu les réseaux RAAM.

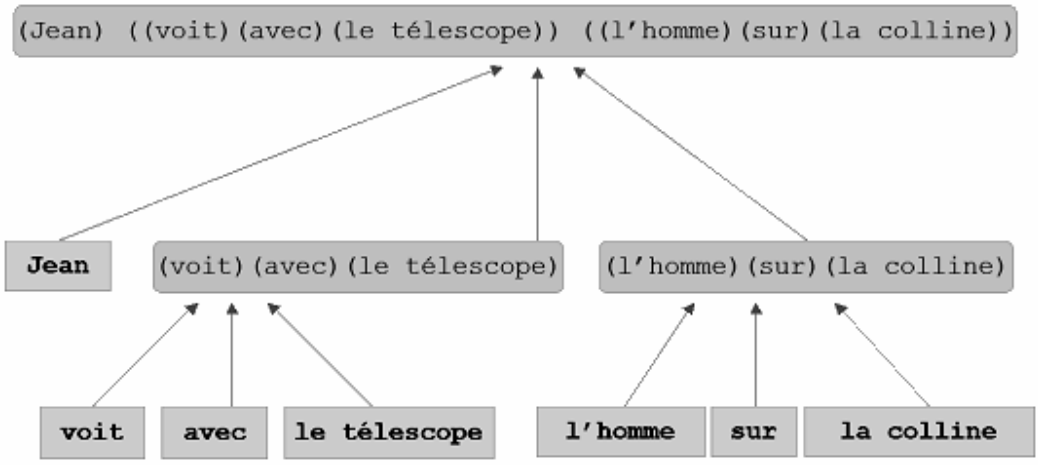
L'idée d'appliquer les représentations générées par les RAAM aux problèmes de compétence logique est évoquée par Bechtel lui-même :

> [I]f problems with compound propositions were included, the ability of networks to handle constituent structure would receive a more challenging test. […] It is not yet clear what kinds of network models will best [convincingly complete the demonstration that logic problems can be solved by processes of pattern recognition using the distributed representations of connectionist networks] but the challenges are not unique to logic and a variety of avenues are being pursued by connectionist investigators. See, for example, Pollack's (1990) RAAM networks and Smolensky's (1990) tensor product representations. (Bechtel, 1994, p. 444)

Les réseaux de Pollack permettent en effet de représenter des structures récursives de taille et de complexité variable par des patterns d'activation de longueur fixe. La méthode de Pollack repose sur l'idée (attribuée à Hinton, 1986) que : « when properly constrained, a connectionist network can develop semantically interpretable representations on its hidden units. » (Pollack, 1990, p. 29) Cette idée exploite ce qui se passe sur la couche cachée d'un réseau « auto-associatif » (de là le « AA » dans « RAAM » : *Recursive **A**uto-**A**ssociative Memory*). La tâche d'un réseau auto-associatif consiste à reproduire sur sa couche de sortie exactement le même pattern d'activation que sur sa couche d'entrée. Si le réseau n'a qu'une couche cachée et que cette couche est composée de moins d'unités que les couches d'entrée et de sortie, on peut considérer que le pattern d'activation généré sur la couche cachée constitue une représentation compressée du pattern présenté en entrée (et reproduit en sortie). Un pattern d'activation *a* présenté sur la couche d'entrée sera propagé jusqu'à la couche cachée, où il sera transformé en pattern d'activation (habituellement) plus petit *c*, puis propagé de là jusqu'à la couche de sortie où il sera retransformé en *a*. On a donc le schéma suivant : $a \rightarrow c \rightarrow a$. Une fois le réseau bien entraîné, on n'a qu'à retenir *c* et, si besoin est, on peut le retransformer en *a* en utilisant la partie supérieure du réseau—la couche cachée et la couche de sortie—comme décodeur ($c \rightarrow a$).

L'idée de Pollack consiste à utiliser ce principe pour encoder des structures arborescentes—ou récursives. Cela lui permet de représenter, par exemple, la compositionalité que l'on retrouve dans les phrases du langage. Par exemple, la phrase « Jean voit avec le télescope l'homme sur la colline —utilisée par Pollack dans une de ses expériences (1990, §3.2.2)—a la structure suivante :

Figure 1 - Structure arborescente de la phrase "Jean voit avec le télescope l'homme sur la colline"

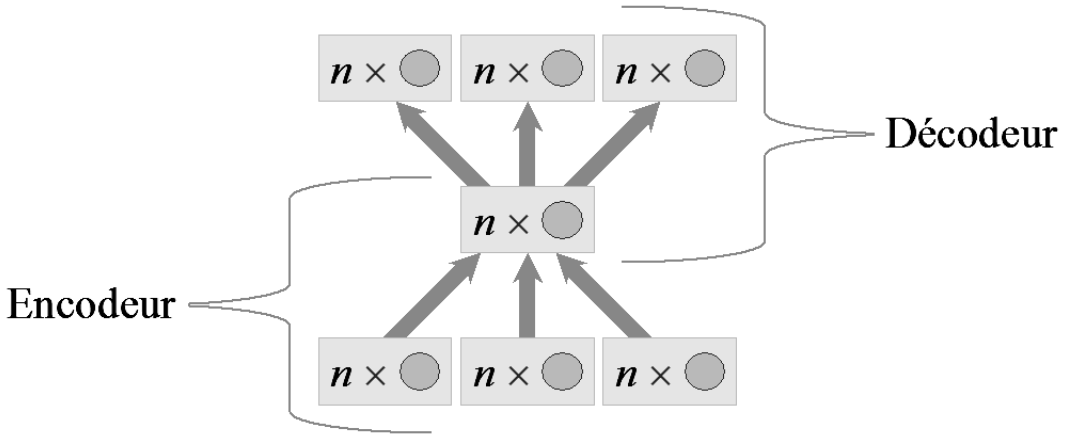


Les composantes atomiques sont représentées en gras, dans les boîtes carrées; les composantes moléculaires dans les boîtes aux coins arrondis. La structure de l'arbre est ternaire[23] : toutes les composantes moléculaires sont formées de trois composantes atomiques. Supposant que nous ayons une représentation de *n* bits pour chacune des composantes atomiques, nous pourrions encoder la phrase complète en un pattern de longueur *n* avec un réseau auto-associatif qui aurait la structure suivante[24] :

[23] Il est aussi possible d'utiliser des RAAM pour encoder des structures binaires (Pollack (1990) le fait) et même quaternaires ou plus.

[24] Les boîtes [n × ❍] représentent des groupes de *n* neurones et les flèches pleines reliant deux de ces groupes représentent des connexions entre chaque neurone du premier groupe et chaque neurone du deuxième groupe.

Figure 2 - Structure d'un réseau RAAM



Ainsi, lorsqu'on présente au réseau sur sa couche d'entrée les représentations pour « voit », « avec » et « le télescope », le pattern d'activation sur la couche cachée peut être interprété comme « (voit)(avec)(le télescope)» et servir ensuite à l'encodage de la phrase complète[25], pour laquelle on obtient alors une « représentation récursive distribuée ».

Remarquons ici que, selon la distinction évoquée par van Gelder, ces représentations font preuve de compositionalité fonctionnelle, mais pas syntaxique. On ne retrouve, dans la représentation finale, aucune instanciation explicite de ses composantes, mais il est néanmoins possible de retrouver ces composantes en utilisant la partie « décodeur » du réseau.

Cette compositionalité fonctionnelle est suffisante pour que les représentations récursives distribuées puissent être utilisées dans certaines tâches cognitives. David Chalmers (1990), par exemple, a entraîné un réseau à transformer des représentations récursives distribuées développées pour des phrases au mode actif en représentations équivalentes pour des phrases au mode passif. Il s'agissait, par exemple, de transformer la phrase « Jean aime Michel », dont l'arbre ternaire est ((Jean)(aime)(Michel)) pour obtenir la phrase « Michel est aimé par Jean », dont l'arbre ternaire est ((Michel) ((est)(aimé)(nil)) ((par)(Jean)(nil))). Le succès de l'expérience de Chalmers (un taux de généralisation de 100% dans certaines conditions) montre déjà que la

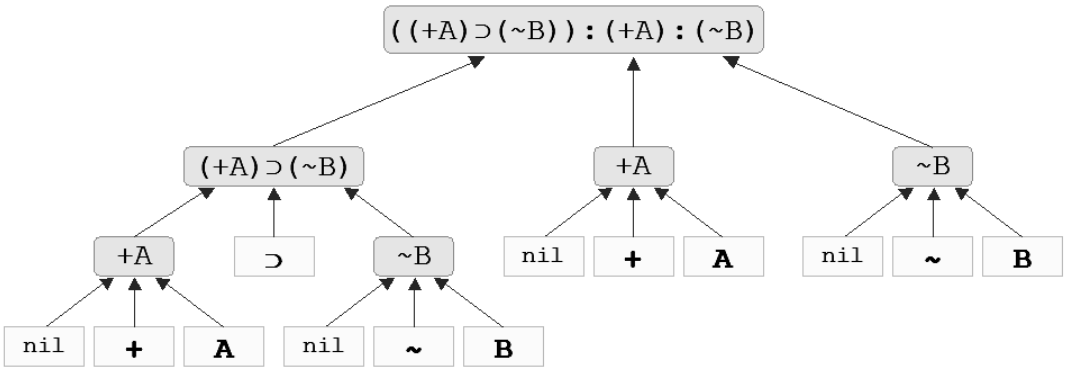
[25] Notons dès maintenant la difficulté posée par le fait que le réseau RAAM, lors de sa phase d'entraînement, cherche à atteindre une « cible en mouvement ». En effet, la représentation compressée que le réseau doit développer pour (Jean)((voit)(avec)(le télescope))((l'homme)(sur)(la colline)) » repose sur les représentations développées pour (voit)(avec)(le télescope) et (l'homme)(sur)(la colline), qui, elles-mêmes, changent au cours de l'entraînement. Cette situation rend l'algorithme d'apprentissage très sensible à certains paramètres, notamment le taux d'apprentissage (η) et le momentum (α) (cf. Pollack, 1990 pour une courte discussion). Pour le momentum, j'ai utilisé $\alpha \leftarrow 0.3 + 0.6 \times (1\text{-taux\_error}^{0.25})$, mis à jour après chaque époque. Pour le taux d'apprentissage, j'ai utilisé $\eta \leftarrow \{\eta/0.99$ si l'erreur diminue depuis plus de 5 époques; 0.99η si l'erreur reste stable ou augmente; η sinon$\}$, mis à jour après chaque époque, avec une valeur initiale de 1, un plancher de 0.01 et un maximum de 5.

compositionalité fonctionnelle pourrait être suffisante pour faire preuve de systématicité. Nous allons maintenant voir comment les représentations récursives distribuées peuvent être utilisées pour les tâches de logique proposées par Bechtel.

Encodage de propositions logiques

Les propositions logiques se prêtent bien à la décomposition en arbres ternaires. Dans le cas des arguments simples—deux prémisses et une conclusion—utilisées dans la première tâche de Bechtel, cette structure est même applicable au niveau des arguments. Par exemple, pour le *modus ponens* « Si A alors non-B, or A, donc non-B » :

Figure 3 - Arbre ternaire pour un *modus ponens*[26]



Pour encoder toutes les propositions nécessaires aux tâches de Bechtel, il faut des opérateurs binaires, des opérateurs unaires et des constantes. Plusieurs méthodes auraient pu être utilisées[27] mais il a été déterminé empiriquement que l'encodage suivant donnait des résultats satisfaisants :

[26] Dans l'arbre ci-dessus, le symbole « + » est utilisé comme opérateur d'affirmation.

[27] Ma méthode d'encodage pour les représentations terminales est similaire à celle de Pollack (1990), en ce que, dans les représentations atomiques binaires, les différentes catégories de symboles sont distinguées par les bits utilisés (i.e. les 5 premiers bits pour les opérateurs binaires, les 3 suivants pour les opérateurs unaires et les 7 derniers pour les constantes). J'ai par la suite découvert la méthode de van Gelder et Niklasson (1994, p. 7-8), qui développent pour les différents symboles des représentations distribuées qui encodent leur catégorie dans une structure arborescente. Par exemple, pour la proposition S: S^D = `S + (Proposition + (Symbol + Nil))`. Cette méthode offre des possibilités intéressantes et aurait mérité d'être appliquée ici.

Tableau 13 - Encodage des composantes atomiques

Opérateurs binaires	⊃	**10001**	000	0000000
	∨	**10010**	000	0000000
	|	**10100**	000	0000000
	&	**11000**	000	0000000
Opérateurs unaires	+	00000	**101**	0000000
	~	00000	**110**	0000000
Constantes	**A**	00000	000	**1000001**
	B	00000	000	**1000010**
	C	00000	000	**1000100**
	D	00000	000	**1001000**
	E	00000	000	**1010000**

Nous avons donc des composantes atomiques de 15 bits. Pour générer des représentations récursives distribuées à partir de celles-ci, il faut un réseau RAAM avec 45 unités sur la couche d'entrée (trois groupes de quinze), 15 unités sur la couche cachée et 45 unités sur la couche de sortie. Pour encoder une implication comme « Si A, alors B », il suffit de présenter au réseau à l'entraînement les représentations binaires pour « A », « ⊃ » et « B » sur les trois groupes de 15 neurones de la couche d'entrée (et de demander au réseau de reproduire ce pattern en sortie). Dans le cas d'une négation comme non-B, on aura NIL (tous des zéros) pour le premier groupe, l'opérateur « ~ » pour le groupe central, et « B » pour le troisième groupe. Les deux figures suivantes illustrent cette méthode :

Figure 4 - Encodage avec opérateur binaire

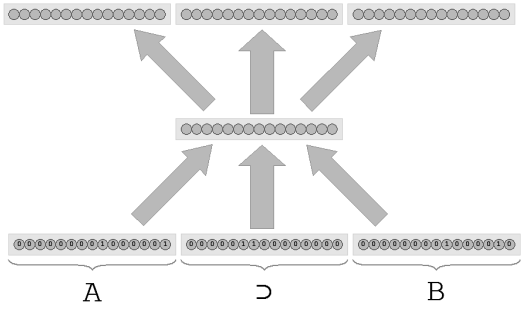
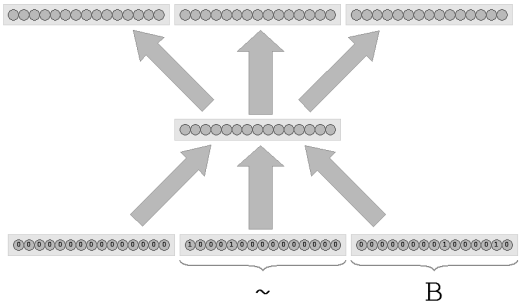


Figure 5 - Encodage avec opérateur unaire



Maintenant que nous avons un aperçu général des RAAM et de la façon dont nous pouvons les utiliser pour l'encodage de propositions logiques, il est temps de se pencher à nouveau sur les tâches de Bechtel.

Première tâche : reconnaissance et évaluation d'arguments

La première tâche de Bechtel se prête bien à l'utilisation de représentations récursives distribuées puisque les arguments eux-mêmes ont une structure ternaire et peuvent être encodés au même titre que les propositions. Il s'agit donc, dans un premier temps, de créer des représentations pour toutes les propositions utilisées dans un argument puis, dans un deuxième temps d'utiliser ces représentations pour développer des représentations d'arguments complets. Finalement, un réseau distinct est entraîné à reconnaître la forme et la validité des arguments ainsi représentés.

J'ai d'abord utilisé cette technique pour entraîner un réseau sur le même ensemble de problèmes que Bechtel, ce qui a donné de très bons résultats (meilleurs que ceux de Bechtel). J'ai ensuite complexifié la tâche en ajoutant à l'ensemble de problèmes des arguments formés à l'aide de propositions de « niveau deux » : dans la mesure où $A \supset B$ est une proposition de niveau un, $(A \vee C) \supset (B \vee D)$ est une proposition de niveau deux. À chacun des niveaux, les opérandes peuvent être niés ou non—par exemple, $\sim(A \vee \sim C) \supset (\sim B \vee D)$. Non seulement le réseau a-t-il appris facilement à reconnaître et à évaluer les arguments ainsi formés, mais ses capacités de généralisation se sont révélées encore meilleures sur ces problèmes.

Encodage et architecture

Un même réseau RAAM est utilisé pour encoder à la fois les propositions de niveau un et les propositions de niveau deux. Ces propositions sont générées de façon combinatoire. Pour générer le niveau un, on prend d'abord les quatre constantes (A, B, C et D) et on leur applique les opérateurs unaires (affirmation et négation), puis les opérateurs binaires ($\supset$, $\vee$ et $|$)[28]. On procède de la même façon pour générer le niveau deux, mais en partant des propositions de niveau un plutôt que des constantes.

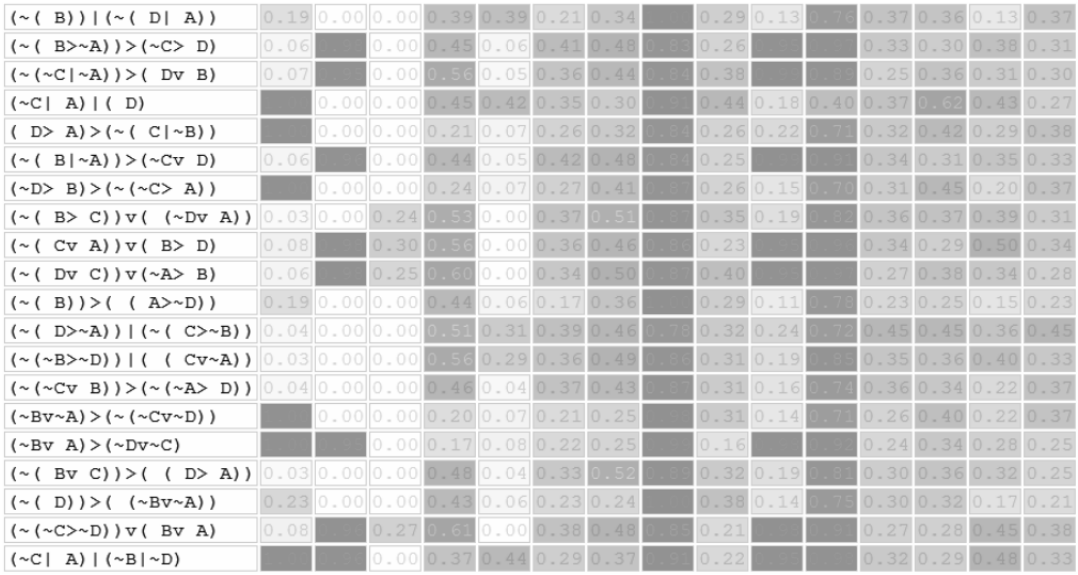
Les propositions sont générées de façon à éviter la répétition de constantes à l'intérieur d'une proposition. Si on s'en tient à des propositions de niveau un, de telles répétitions ne seraient pas utiles puisque aucun des schémas d'arguments utilisés par Bechtel ne fait appel à des propositions comme $A \supset A$, $B \vee B$ ou $C \mid C$. C'est toutefois lorsqu'on fait intervenir des

[28] Notons ici une petite différence par rapport à Bechtel dans le traitement des doubles négations. Celles-ci peuvent se présenter lorsque des constantes niées sont utilisées pour générer des arguments à partir de schémas qui comportent la négation d'une constante. Par exemple, si on insère les constantes ~A et B dans le schéma « $p \vee q$, or $\sim p$, donc q », on devrait obtenir l'argument : « ~A ∨ B, or ~~A, donc B ». La méthode d'encodage utilisée par Bechtel ne permet pas d'exprimer ces doubles négations, alors celles-ci sont transformées en simples affirmations. Les doubles négations ne posent toutefois aucun problème lorsque vient le temps de développer des représentations récursives distribuées, puisque ~~A correspond simplement à l'arbre ternaire (NIL)(~)((NIL)(~)(A)).

propositions de niveau deux que cette contrainte prend toute son importance, puisqu'elle nous évite de générer des contradictions comme (A & B) ⊃ ~(A ∨ C). Elle a aussi comme effet secondaire d'exclure des propositions légitimes telles que (A & B) ⊃ (B & C), mais c'est un petit prix à payer pour la simplicité du processus.

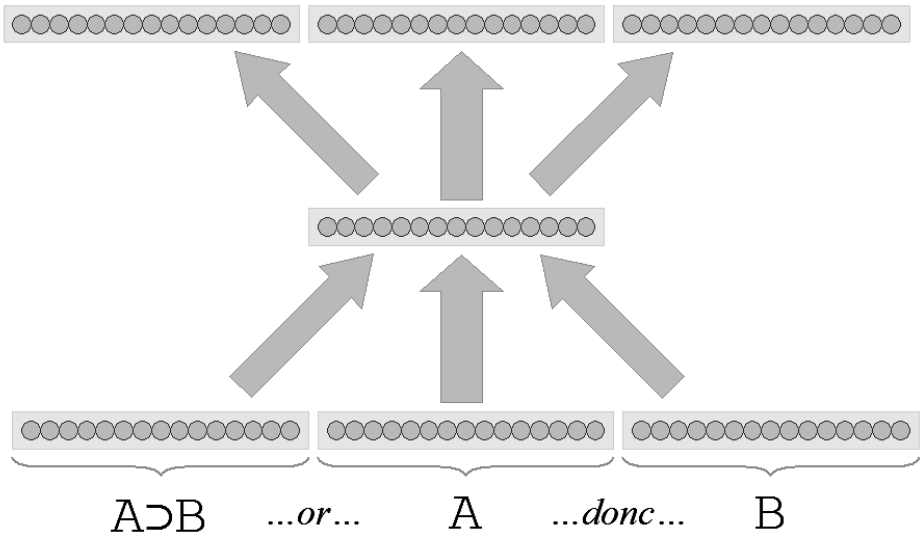
Au bout du compte, ce procédé génère 126 168 propositions différentes. Un réseau RAAM permet de développer pour chacune d'elles une représentation distribuée sur 15 unités. Chaque proposition correspond ainsi à un point dans un espace à 15 dimensions (cette interprétation nous servira plus tard). La figure suivante montre quelques exemples de représentations développées par le réseau RAAM. Chaque colonne représente une unité de la couche cachée. Plus la valeur d'activation d'une unité est élevée pour une proposition, plus la cellule correspondante est foncée :

Figure 6 - Exemples de représentations pour différentes propositions



Une fois les représentations générées pour toutes les propositions, c'est au tour des arguments, qui sont générés à partir des schémas d'arguments de Bechtel (cf. Tableau 1, ci-dessus). Les deux versions de la tâche sont traitées séparément. On génère d'abord les 576 arguments utilisés par Bechtel, puis on entraîne un RAAM pour développer des représentations pour ceux-ci. Les deux premiers groupes de 15 unités reçoivent les prémisses et le troisième reçoit la conclusion :

Figure 7 - Encodage d'un argument



La technique est la même pour la version de l'expérience qui utilise des propositions de niveau deux en plus des propositions de niveau un, sauf qu'il s'agit alors d'encoder 55 872 arguments plutôt que 576.

Une fois tous les arguments encodés, reste à entraîner des réseaux (un pour chaque version) à reconnaître leur forme et à évaluer leur validité. La même architecture que Bechtel est utilisée, sauf que la couche d'entrée comporte 15 unités plutôt que 14. On a donc deux couches cachées à 10 unités, et une couche de sortie à trois unités—deux pour indiquer la forme de l'argument et une pour sa validité.

L'entraînement se passe toujours de la même façon. L'ensemble de problèmes est divisé en trois; le réseau est entraîné sur le premier sous-ensemble (E_1); la généralisation est testée sur le deuxième sous-ensemble (E_2); le réseau est encore entraîné sur les deux premiers sous-ensembles (E_1+E_2) et la généralisation est finalement testée sur le troisième sous-ensemble (E_3).

Résultats

Le tableau suivant présente les résultats des deux versions de l'expérience—niveau un (N1) et niveaux un et deux (N1 + N2)—comparés à ceux obtenus par Bechtel pour le niveau un (déjà présentés dans le Tableau 5).

Tableau 14 - Résultats pour la première tâche de Bechtel

	Bechtel (N1)		Payette (N1)		Payette (N1 + N2)	
Entraînement E_1	3000		383		153	
Test E_1	192 / 192	100%	192 / 192	100%	18 624 / 18 624	100%
Test E_2	146 / 192	76%	184 / 192	97,4%	18 605 / 18 624	99,9%
Entraînement E_1+E_2	5000		40		132	
Test E_1+E_2	380 / 384	98,9%	384 / 384	100%	37 248 / 37 248	100%
Test E_3	161 / 192	83,9%	187 / 192	95,8%	18 620 / 18 624	99,98%

On remarquera que le réseau N1 n'a eu besoin que de 383 époques—comparativement à 3000 pour Bechtel—pour atteindre 100% de succès sur le premier ensemble d'entraînement. La généralisation à E_2 est aussi bien meilleure (97,4% vs. 76%).

Comment expliquer cette différence de performance? Pour ce qui est du nombre d'époques d'apprentissage requises, il est possible que les paramètres utilisés pour l'algorithme de rétropropagation[29], de même que l'unité supplémentaire dans la couche d'entrée, aient joué un rôle mineur. Il est toutefois plus vraisemblable que les représentations utilisées soient responsables. On peut supposer que, lors de leur apprentissage, les réseaux RAAM ont saisi quelque chose de la structure des propositions qui facilite la reconnaissance des formes d'arguments : leur compositionalité fonctionnelle. Le même facteur joue probablement un rôle dans la performance du réseau N1 + N2, qui est presque parfaite. Notons toutefois que, pour ce dernier, le nombre d'arguments dans chaque sous-ensemble d'entraînement est beaucoup plus élevé, et que le réseau bénéficie ainsi de beaucoup plus d'exemples lors de chaque époque d'apprentissage.

Deuxième tâche : déduction naturelle

Lorsque Bechtel discute des limites de son modèle de la déduction naturelle, l'un des premiers points qu'il mentionne est que : « Real natural deduction systems allow for embedding of compound propositions within the components of a proposition, as in ((A & B) ∨ C). » (1994, p. 456) C'est exactement ce que les représentations récursives distribuées nous permettent de faire. Ainsi, par exemple, le schéma I de Bechtel peut servir à générer la dérivation suivante, qui fait intervenir des propositions de « niveau deux » :

[29] Bechtel ne spécifie pas les paramètres utilisés. De mon côté, j'ai utilisé la même technique que pour entraîner les réseaux RAAM, qui fait varier le momentum et le taux d'apprentissage en fonction de l'erreur. Cf. note 25, plus haut.

Tableau 15 - Exemple de dérivation avec propositions de niveau 2

Schéma I			Dérivation	Opération
Prémisse	1.	*p* ∨ *q*	~A ∨ **(B & ~E)**	
Prémisse	2.	*q* ⊃ *r*	**(B & ~E)** ⊃ C	
Prémisse	3.	~*p*	~~A	
Inférence	4.	*q*	**(B & ~E)**	:1,3 ∨-elim
Inférence	5.	*r*	C	:2,4 ⊃-elim
Conclusion	6.	*r* ∨ *s*	C ∨ D	:5 ∨-intro

Pour cette tâche aussi, l'idée était de reproduire d'abord l'expérience de Bechtel avant d'explorer la possibilité de faire intervenir des propositions de niveau deux. Notons toutefois que seule la variante sans négations a été reproduite; les contraintes utilisées par Bechtel pour la version avec négations rendaient—selon moi—l'exercice moins intéressant. Les négations ont plutôt été introduites, sans appliquer de contraintes, dans la version avec propositions de niveau deux.

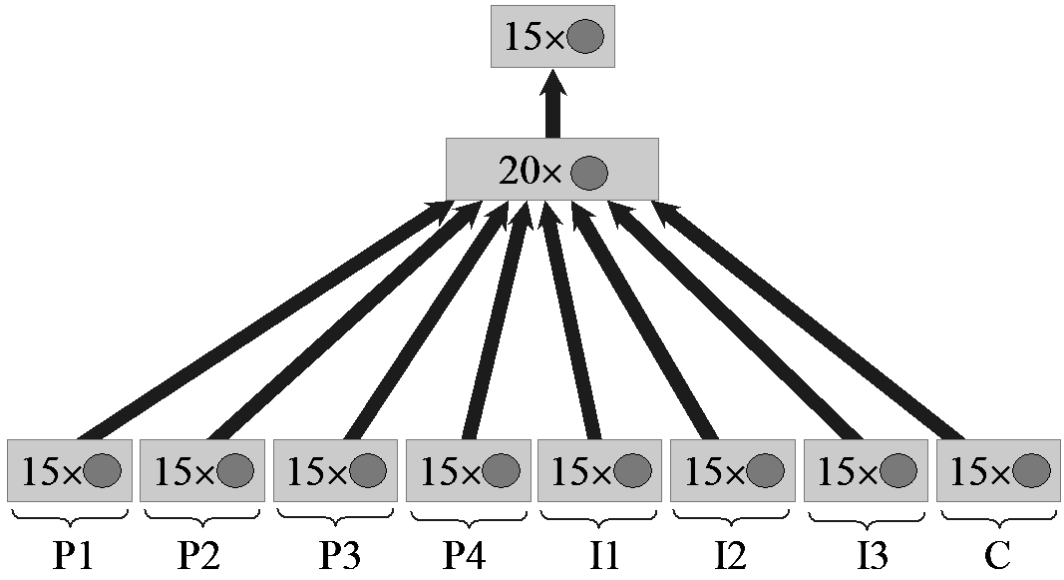
L'introduction de propositions de niveau deux entraîne certaines difficultés sur le plan computationnel. Nous verrons que les résultats obtenus laissent place à amélioration, mais qu'ils laissent néanmoins entrevoir les possibilités ouvertes par les représentations récursives distribuées pour ce genre de problème.

Encodage et architecture

L'encodage des propositions se fait de la même façon que dans la première tâche. Dans la version « niveau un, sans négations », il suffit de troquer l'opérateur | pour l'opérateur &—et bien sûr de ne pas utiliser l'opérateur ~. Dans la version « niveau deux, avec négations », il nous faut toutefois rajouter une cinquième constante (E), puisque certains schémas de dérivation font déjà appel à quatre constantes et qu'il nous faut une constante supplémentaire pour pouvoir introduire le niveau deux. L'ajout de cette constante allonge sensiblement le temps d'entraînement du réseau RAAM, puisqu'on se retrouve alors avec 547 230 propositions possibles.

Côté architecture, on a sensiblement la même chose que Bechtel, sauf que chacun des groupes comporte 15 unités plutôt que 13 :

Figure 8 - Architecture du réseau de déduction naturelle



Pour la version « niveau deux, avec négations », on utilisera (comme le faisait Bechtel pour sa version avec négations) 40 unités sur la couche cachée.

L'utilisation de représentations récursives distribuées introduit toutefois une difficulté à laquelle Bechtel n'a pas eu à faire face. Dans l'expérience de Bechtel, les réponses attendues du réseau sont des représentations binaires. Pour s'assurer qu'il en soit bien ainsi, il suffit d'utiliser un seuil : toute unité dont l'activation est à inférieure à 0.5 est considérée comme ayant une activation de 0, toute autre unité comme ayant une activation de 1. Ici, par contre, la réponse attendue est une représentation distribuée et l'activation des unités de la couche doit donc être beaucoup plus précise, puisque la partition de l'espace de représentations est beaucoup plus fine. Nous ne pouvons toutefois pas nous attendre à ce que l'activation de ces unités corresponde *exactement* à la représentation attendue. Pour vérifier que le réseau a bien appris, il nous faut trouver une façon d'associer la réponse du réseau à une proposition particulière. La méthode retenue consiste à utiliser la distance euclidienne dans l'espace des représentations : la proposition qui est la plus « proche » du point désigné par l'activation de la couche de sortie dans cet espace à 15 dimensions est considérée comme la réponse du réseau.

Résultats

Pour la version « niveau un, sans négations », les dérivations ont été générées de la même façon que l'avait fait Bechtel, avec trois permutations des prémisses pour chaque schéma de I à XIV. On arrive à un total de 1 008 dérivations impliquant 2 736 inférences. Trois quarts de ces dérivations (E_1) servent à l'entraînement du réseau et l'autre quart (E_2) sert à tester la généralisation[30]. La généralisation a aussi été testée ici avec un troisième ensemble, comprenant 1 052 inférences générées à partir des schémas XV-XX, schémas sur lesquels le réseau n'a reçu aucun entraînement. Les résultats obtenus sont les suivants :

	Bechtel (N1,↛)		**Payette (N1,↛)**	
Entraînement E_1	500		2500	
Test E_1	1800 / 1800	100%	1948 / 2060	94,56%
Test E_2	767 / 936	81,9% (90,5% WTA)	627 / 676	92,75%
Test E_3	N/A	N/A	661 / 1052	57,38%

Le réseau a eu besoin de 2 500 époques pour atteindre un taux de succès de 94,56% sur l'ensemble d'entraînement. Cela peut sembler moins bon que les résultats de Bechtel, mais il faut se rappeler que les réponses du réseau doivent être beaucoup plus précises. Les représentations binaires sont disposées dans l'espace de façon orthogonale. Les représentations distribuées, elles, forment plutôt des groupements de similarité—des « clusters »—assez denses, que le réseau doit partitionner finement lors de son apprentissage. Cette situation a toutefois ses avantages lorsque vient le temps de généraliser, comme le montre la performance de 92,75% sur l'ensemble E_2. Le test sur E_3 s'est aussi révélé assez concluant, le réseau arrivant à faire correctement 57,38% des inférences pour des schémas de dérivation jamais rencontrés auparavant.

Dans la deuxième version de l'expérience—niveau deux, avec négations—on génère autant de dérivations qu'il est possible de le faire avec nos cinq constantes[31] et les vingt schémas proposés par Bechtel[32]. Cela nous donne 2 426 400 dérivations, comprenant 17 294 400 inférences. C'est énorme. Il serait impraticable d'entraîner un réseau sur les trois quarts de ces inférences. Il a plutôt été décidé de choisir au hasard un certain nombre de dérivations, de façon à former des ensembles d'entraînement et de test de tailles comparables à ceux qui ont été utilisés pour la première variante. On a donc 756 dérivations, comprenant 2 141 inférences, dans E_1—ce

[30] Notons que le nombre d'inférences à faire dans chaque ensemble d'entraînement dépend du processus de séparation aléatoire des dérivations, puisque ce ne sont pas toutes les dérivations qui ont le même nombre d'étapes. C'est ce qui explique la différence avec Bechtel dans la répartition des inférences entre E_1 et E_2.
[31] Ici aussi, on évite les répétitions de constantes.
[32] Avec trois permutations de prémisses.

qui représente seulement 0,012% du total des inférences. Dans E_2, on a 252 dérivations et 704 inférences—0,004% du total des inférences. Dans E_3, on a 1 000 dérivations et 2 784 inférences—0,016% du total des inférences. Les dérivations ont été choisies au hasard dans chacun des ensembles d'entraînement et aucune contrainte n'a été imposée pour s'assurer que tous les schémas étaient représentés, mais les chances qu'un schéma ne soit pas représenté sont négligeables (pour E_1, par exemple : 0,000001%). Après 500 époques d'entraînement, les résultats sont les suivants :

	Payette (N1-N2,-)	
Entraînement E_1	500	
Test E_1	458 / 2 141	21,39%
Test E_2	88 / 704	12,50%
Test E_3	249 / 2784	8,94%

La tâche est manifestement plus difficile pour le réseau. On se rappellera que le nombre de propositions possibles est 547 230 et que l'espace des représentations est donc fort densément peuplé. Le réseau n'arrive à faire correctement qu'une inférence sur cinq. Lorsque vient le temps de généraliser, son taux de succès tombe à 12,5%. C'est moins que ce qui aurait été souhaité, mais le réseau a quand même appris quelque chose. En effet, la chance ne lui aurait permit de faire correctement qu'une inférence sur 547 230—autant dire aucune! Le taux est à peine plus bas pour E_3 (8,94%) sur des dérivations qui suivent des schémas inconnus du réseau.

Plusieurs stratégies sont possibles pour améliorer ces performances. Trois axes d'interventions se présentent d'emblée : la composition des ensembles d'entraînement, l'architecture du réseau et les représentations distribuées elles-mêmes. En ce qui concerne les ensembles d'entraînement, on pourrait à la fois augmenter la taille de E_1 et s'assurer que son contenu soit plus représentatif du problème général—que chaque schéma de dérivation soit également représenté dans l'ensemble. Ensuite, on peut intervenir sur l'architecture du réseau : augmenter le nombre de neurones dans la couche cachée ou même le nombre de couches cachées. On pourrait aussi augmenter la taille des représentations distribuées. Quinze unités suffisaient pour encoder des propositions à quatre constantes, mais l'augmentation du nombre de propositions due à l'ajout d'une cinquième constante justifie probablement l'ajout de quelques dimensions à l'espace des représentations—ce qui permettraient d'augmenter la distance entre les représentations dans cet espace et donc de faciliter la tâche du réseau qui doit les distinguer. Il semble d'ailleurs raisonnable de supposer que, dans un cerveau humain, de telles propositions sont encodées sur plus de quinze neurones. Finalement, il serait envisageable d'augmenter le

temps d'entraînement des réseaux RAAM, temps qui avait été limité à 1 000 époques pour des raisons pratiques. Notons que toutes ces interventions sont relativement simples à implémenter, mais que chacune d'entre elles allonge considérablement le temps requis pour entraîner et tester le réseau[33].

Conclusion

Dans ce qui précède, nous avons abordé deux conceptions rivales de l'esprit. Pour le classicisme, la cognition consiste essentiellement à effectuer des opérations logiques portant sur des symboles discrets, un peu comme le fait un ordinateur. Si on arrive à comprendre comment fonctionnent ces opérations qui constituent le « langage de la pensée » on aura expliqué l'esprit, et par le fait même ses caractéristiques essentielles comme la productivité et la systématicité. Cela ne veut pas dire qu'il est inutile de chercher à comprendre le fonctionnement du cerveau mais, au dire des classicistes, cette tâche ne peut nous éclairer que sur la façon dont sont implémentées les lois psychologiques, pas sur les lois elles-mêmes. Elle ne relève donc pas de la psychologie.

Les connexionnistes, eux, privilégient plutôt une approche ascendante, où c'est l'étude des propriétés computationnelles des réseaux de neurones qui vient en premier. Les lois psychologiques (s'il y en a) et les propriétés comme la productivité et la systématicité émergent des interactions de multiples neurones qui travaillent en parallèle. Bechtel, lui, va encore plus loin, en affirmant qu'il faut aussi faire intervenir l'interaction entre le cerveau et l'environnement pour expliquer des propriétés telles que la systématicité et la productivité.

Fodor et Pylyshyn souhaitent couper court aux prétentions des connexionnistes en soutenant qu'il est impossible pour eux de rendre compte de la productivité et de la systématicité de la pensée humaine. Dans le cas de la productivité, leur argument repose sur le fait que l'ajout de mémoire à un réseau connexionniste modifie sa structure computationnelle, ce qui mine les capacités explicatives du réseau. Dans le cas de la systématicité, Fodor et Pylyshyn insistent pour dire que celle-ci est impossible sans compositionalité syntaxique. Cette compositionalité est à la fois une caractéristique essentielle du classicisme et incompatible avec le connexionnisme.

La réponse de Bechtel, c'est que la productivité et la systématicité ne sont pas des caractéristiques d'un esprit isolé, mais plutôt des propriétés du système formé par le couplage du

[33] Les expériences décrites ci-dessus ont été menées à l'aide du langage C++, en utilisant le compilateur GCC et la base de données MySQL (et son API pour C). Aucune autre librairie externe. Le code est disponible sur demande auprès de nicolaspayette@gmail.com.

cerveau et de l'environnement. Nul besoin de pouvoir ajouter de la mémoire ou d'implémenter la compositionalité syntaxique si on a un réseau qui peut manipuler des symboles externes. Il suffit au réseau de savoir reconnaître les problèmes qui lui sont posés et de pouvoir manipuler l'environnement de façon à emmener tranquillement ces problèmes à leur résolution. Pour appuyer ses dires, Bechtel nous présente des réseaux qui arrivent à reconnaître la forme et la validité de certains arguments logiques, et d'autres qui arrivent à compléter des dérivations logiques en plusieurs étapes. Par leur habilité à généraliser, ces réseaux font preuve de capacités qui ressemblent à celles qui sont exigées par Fodor et Pylyshyn. Ils arrivent identifier des arguments et à résoudre des dérivations dont les composantes (et même les formes!) n'ont jamais été rencontrées auparavant, montrant que leur capacité à traiter ces cas nouveaux est reliée à leur capacité à traiter ceux de leur ensemble d'entraînement. Il semble à Bechtel (et à moi aussi) que cela constitue une forme de systématicité tout à fait satisfaisante. En ce qui concerne la productivité, toutefois, la question est plus délicate. En quoi consisterait, pour un réseau de neurones artificiels, une capacité de généralisation infinie? Je ne suis pas sûr de pouvoir donner une réponse à cette question, mais je crois pouvoir poser le problème autrement en faisant appel aux représentations récursives distribuées.

Pollack concède à Fodor et Pylyshyn que la compositionalité des représentations est importante pour la cognition mais il nous montre aussi que cette compositionalité n'a pas besoin d'être syntaxique, c'est-à-dire que les représentations atomiques n'ont pas besoin de se retrouver *telles quelles* dans les représentations moléculaires. Les représentations récursives distribuées construites par les réseaux RAAM font plutôt preuve de compositionalité fonctionnelle, en ce que les constituants atomiques peuvent être *reconstruits* à partir des représentations moléculaires. Comme le montrent Pollack (1990) et Chalmers (1990), ces représentations peuvent subir des transformations qui dépendent de leur structure compositionnelle, mais elles peuvent aussi être traitées de façons holistique. C'est de cette façon que je les ai utilisées pour les tâches de Bechtel, obtenant ainsi une performance nettement améliorée. De plus, elles m'ont permis de soumettre aux réseaux des problèmes faisant intervenir des propositions de différents niveaux de complexité. Comme le montrent les résultats de l'expérience de déduction naturelle lorsqu'on introduit des propositions de « niveau deux », il y a un prix cognitif à payer pour cette complexité. Notons toutefois que nous avons ici affaire à une dégradation progressive de la performance du réseau et non à une incapacité fondamentale à traiter la complexité. Il serait, en théorie, possible d'utiliser un même réseau RAAM pour encoder des propositions de complexité toujours plus grande. La qualité des représentations ainsi développées seraient de moins en

moins bonne et celles-ci seraient de plus en plus difficile à traiter pour les réseaux[34], mais aucun changement ne serait exigé dans la structure computationnelle du réseau. Est-ce que cela constitue une capacité productive au sens où l'entendent Fodor et Pylyshyn? Je n'en suis pas sûr, mais mon cerveau a de la difficulté à traiter les concepts faisant appel à la notion d'infini. Je crois toutefois que cela constitue une bonne approximation de nos capacités cognitives actuelles. Notre cerveau est doté d'un nombre fini de neurones et, bien que nous soyons capables de traiter des propositions très complexes, notre performance se dégrade à mesure que cette complexité augmente. Qu'une théorie psychologique arrive à rendre compte de nos failles aussi bien que de nos forces me semble être un avantage indéniable.

Le travail présenté ici tente de tirer profit à la fois de l'approche plus externaliste de Bechtel—qui se penche sur la relation entre l'esprit et le monde—et l'approche plus internaliste de Pollack—qui se penche sur la structure de nos représentations du monde. La combinaison de ces deux approches n'est pas un choix innocent. Les systèmes cognitifs que l'on retrouve dans la nature sont le fruit d'un processus—l'évolution—qui tire profit de tout ce qu'il a à sa disposition pour adapter l'organisme à son environnement, que ce soit les structures cérébrales préexistantes ou encore l'environnement lui-même. La compétence logique, aussi tentant soit-il de la mettre sur un piédestal, n'échappe pas à ces contraintes. Les lacunes dans notre performance sont là pour en témoigner. Il est vrai que l'hypothèse du langage de la pensée a quelque chose de séduisant. Elle est simple et harmonieuse, comme les théories que l'on rencontre parfois en physique. Ou comme la grammaire universelle postulée par Chomsky. Mais il semble malheureusement que l'esprit ne fonctionne pas comme ça, et que la barbe de la psychologie doive rester trop touffue pour le rasoir d'Occam. Plus nous en apprenons sur le cerveau et l'esprit et plus il semble que l'hypothèse du langage de la pensée doive être rangée comme « a case of misprojecting the categories of language back onto the activities of the brain too enthusiastically. » (Dennett, 1991, p. 365)

Bibliographie

Bechtel, W. 1996a. «What should a connectionist philosophy of science look like?». In *The Churchland and their Critics*, R. N. McCauley. Oxford: Basil Blackwell. En ligne. <http://mechanism.ucsd.edu/~bill/research/CHURCHLA.pdf>.

[34] Quoique, selon moi, des tâches comme la reconnaissance des formes d'arguments ne seraient pas affectées; mais c'est là une affirmation qui demanderait à être vérifiée.

Bechtel, William. 1994. «Natural Deduction in Connectionist Systems». *Synthese*. vol. 101, no 3, p. 433-463.

Bechtel, William. 1996b. «What Knowledge Must Be in the Head that We Might Acquire Language?». In *Communicating meaning: The evolution and development of langauge*, B. Velichkovsky et D. M. Rumbaugh. Hillsdale, NJ: Lawrence Erlbaum Associates. En ligne. <http://mechanism.ucsd.edu/~bill/research/LANGUAGE.pdf>.

Bechtel, William, et Adele A. Abrahamsen. 1991. *Connectionism and the mind : an introduction to parallel processing in networks*. Cambridge, Mass., USA: B. Blackwell, 349 p.

Bechtel, William, et Adele A. Abrahamsen. 1993. *Le connexionnisme et l'esprit : introduction au traitement parallèle par réseaux*. Paris: Éditions La Découverte, 365 p.

Bechtel, William, et Adele A. Abrahamsen. 2002. *Connectionism and the mind : parallel processing, dynamics, and evolution in networks*, 2e édition. Malden, Mass. ; Oxford, England: Blackwell, 406 p.

Bennett, M. R., et P. M. S. Hacker. 2003. *Philosophical Foundations of Neuroscience*: Blackwell Publishing.

Berkeley, I. S. N., M. R. W. Dawson, D. A. Medler, D. P. Schopflocher et L. Hornsby. 1995. «Density plots of hidden value activations reveal interpretable bands». *Connection Science*. vol. 7, p. 167-186.

Chalmers, David J. 1990. «Synctatic Transformations on Distributed Representations». *Connection Science*. vol. 2, no 1-2, p. 53-62.

Chomsky, Noam. 1968. *Language and Mind*. New York: Harcourt, Brace and World.

Christiansen, M. H. 1992. «The (non) necessity of recursion in natural language processing». In *Proceedings of the 14th Annual Conference of the Cognitive Science Society*, p. 665-670. Hillsdale, NJ: Erlbaum.

Clark, Andy. 1997. *Being There: Putting Brain, Body and World Together Again*. Cambridge, MA: MIT Press.

Dawson, M. R. W., D. A. Medler et I. S. N. Berkeley. 1997. «PDF models can provide models that are not mere implementations of classical theories». *Philosophical Psychology*. vol. 10, p. 25-40.

Dennett, Daniel C. 1991. *Consciousness explained*, 1st. Boston: Little, Brown and Co., xiii, 511 p. p.

Dummett, M. 1982. «Realism». *Synthese*. vol. 52, no 1, p. 55-112.

Fodor, J., et B. P. McLaughlin. 1990. «Connectionism and the problem of systematicity: why Smolensky's solution doesn't work». *Cognition*. vol. 35, no 2, p. 183-204.

Fodor, Jerry A. 1975. *The language of thought*. New York: Crowell, 214 p.

Fodor, Jerry A., et Zenon W. Pylyshyn. 1988. «Connectionism and cognitive architecture: A critical analysis». *Cognition*. vol. 28, no 1-2, p. 3-71. En ligne. <http://www.sciencedirect.com/science/article/B6T24-45WHVB2-60/2/2d44ef36e5d784774ca8b5f90231aa3e >.

Hinton, G. E. 1986. «Learning Distributed Representations of Concepts». In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, p. 1-12. Amherst, MA.

Kern, L.H., H.L Mirels et V.G. Hinshaw. 1983. «Scientists' understanding of propositional logic: an experimental investigation». *Social Studies of Science*. vol. 13, p. 131-146.

McClelland, J. L., et D. E. Rumelhart. 1988. *Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises*, no MIT Press. Cambridge, MA.

McClelland, J. L., D. E. Rumelhart et le groupe de recherche PDP. 1986. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 2: Psychological and Biological Models*. Cambridge, Massachusset: MIT Press.

McCulloch, W. S., et W. Pitts. 1943. «A logical calculus of the ideas immanent in nervous activity». *Bulletin of Mathematical Biophysics*. vol. 5, p. 115-133.

Newell, A., et H. A. Simon. 1956. «The logic theory machine». *IRE Transactions on Information Theory*. vol. 3, p. 61-79.

Niklasson, L. F., et T. Van Gelder. 1994. «On being systematically connectionist». *Mind & Language*. vol. 9, no 3, p. 288-302.

Panaccio, Claude. 2004. *Ockham on concepts*. Coll. «Ashgate studies in medieval philosophy». Aldershot, Hants, Angleterre: Ashgate Pub., xi, 197 p.

Poirier, Pierre, et Nicolas Payette. 2007. «Les gardiens du bon usage». *Philosophiques*. vol. 34, no 1.

Pollack, Jordan B. 1989. «Implications of Recursive Distributed Representations». *Advances in Neural Information Processing Systems*. vol. 1, p. 527-536.

Pollack, Jordan B. 1990. «Recursive distributed representations». *Artificial Intelligence*. vol. 46, no 1-2, p. 77-105. En ligne. <http://www.sciencedirect.com/science/article/B6TYF-47YRKG9-38/2/fa77a24cc9dcd706d88c297cdbd7ecb8 >.

Rumelhart, D. E., J. L. McClelland et le groupe de recherche PDP. 1986. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1 : Foundations*. Cambridge, Massachusetts: MIT Press.

Searle, John R. 1980. «Minds, brains, programs». *Behavioral and Brain Sciences*. vol. 3, p. 417-424.

Smolensky, P. 1990. «Tensor Product Variable Binding and the Representation of Symbolic Structures in Connectionist Systems». *Artificial Intelligence*. vol. 46, p. 159-216.

van Gelder, Tim. 1990. «Compositionality: A connectionist variation on a classical theme». *Cognitive Science*. vol. 14, p. 355-384.

van Gelder, Tim, et Lars Niklasson. 1994. «Classicalism and Cognitive Architecture». In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*.

Waskan, J., et W. Bechtel. 1997. «Directions in connectionist research: Tractable computations without syntactically structured representations». *Metaphilosophy*. vol. 28, p. 31-62.

Wason, P. C. 1966. «Reasoning». In *New Horizons in Psychology I*, B. M. Foss. Harmandsworth: Penguin.