Constructions géométriques formelles en CAO

Pascal Schreck, Caroline Essert-Villard

Laboratoire des Sciences de l'Image, de l'Informatique et de la Télédétection
UMR CNRS 7005 – Université Louis Pasteur
Pôle API, Boulevard Sébastien Brant – F-67400 Illkirch, France

{schreck,essert}@dpt-info.u-strasbg.fr

Résumé : *Cet article présente le cadre des constructions géométriques formelles dans le domaine de la résolution de contraintes géométriques en CAO. Les méthodes combinatoires proposées par d'autres équipes sont présentées et comparées à une approche formelle comme la nôtre.*

1 Introduction

La définition d'objets à l'aide d'esquisses cotées est l'un des objectifs du dessin industriel. Un des apports de l'informatique en CAO est de permettre de calculer effectivement et de manipuler un objet ainsi défini par un système de *contraintes dimensionnelles*. Il existe des logiciels industriels qui permettent de résoudre de telles contraintes, mais d'une part les méthodes employées ne sont pas divulguées et, d'autre part ils ont chacuns des inconvénients spécifiques. C'est pourquoi la modélisation par contraintes reste un enjeu important en CAO.

Les techniques qui sont les plus connues dans le milieu académique, comme sans doute dans les milieux industriels, sont essentiellement combinatoires et numériques. Des procédés permettant de réorganiser le système de contraintes, comme la propagation de contraintes qui réalise une triangulation par blocs, sont utilisés conjointement avec des méthodes permettant de décomposer les systèmes de contraintes. Lorsque les sous-systèmes obtenus ne sont pas trivialement solubles, ou que la recomposition des systèmes n'est pas immédiate, des méthodes numériques comme la méthode de Newton-Raphson ou la méthode homotopique sont employées pour effectuer la résolution.

Un autre point de vue, consiste à voir dans ces esquisses cotées des problèmes de construction géométrique comme ceux que l'on rencontre dans les cours de géométrie de type Lycée/Collège. Ce point de vue est à l'origine de quelques travaux en EIAO dont l'objectif est de résoudre *formellement* des problèmes de construction. Formellement est ici à prendre au sens de *calcul formel* : un énoncé est donné sous forme littéral et la solution *exacte* du problème consiste en un procédé de construction général permettant de décrire toutes les solutions dans toutes les situations. Ce procédé est habituellement nommé *programme* ou *plan de construction*.

Cette idée appliquée à la CAO, permet lorsque l'on abstrait les valeurs des cotes d'avoir des *figures paramétrées* où la modification des valeurs des cotes entraîne instantanément la modification de la figure solution selectionnée. Ce point de vue a été initié par B. Brüderlin et surtout par B. Aldefeld à la fin des années 80. Avec notre expérience sur les constructions géométriques dans un cadre EIAO, nous avons repris cette approche pour l'adapter à l'invariance par déplacement qui est une des caractéristiques essentielles des problèmes de CAO. Cet article montre les avantages qu'il y a à suivre une approche formelle comme celle développée dans notre équipe.

La suite de cet article s'organise ainsi : la section 2 présente la terminologie des constructions géométriques formelles, suivie d'un survol des approches classiques de résolution de contraintes géométriques en section 3. Nous montrons comment tirer parti de l'approche formelle dans la section 4, et nous concluons en section 5.

2 Point de vue formel

Rappelons qu'un système formel est constitué de la donnée d'une signature, de connecteurs logiques pour faire des formules, d'axiomes et de règles d'inférence permettant de trouver des théorèmes. Pour qu'un

système formel ait un interêt pratique, on doit ajouter une fonction d'interprétation qui donne un sens aux formules de la théorie formelle. Dans le cadre des constructions géométriques, on doit résoudre des systèmes de contraintes. Ceci revient à chercher des preuves constructives de théorème d'existence. La sémantique du cadre formel doit donc être particulièrement précise.

Univers géométrique. Dans la suite, nous appelons univers géométrique une Σ-algèbre donnée où Σ désigne une signature hétérogène $\Sigma = (S, F, P)$ avec S ensemble de symboles de *sortes*, F ensemble de symboles fonctionnels et P ensemble de symboles prédicatifs. Si X est un ensemble de variables typées, on note $T(S, X)$ l'ensemble des termes bien formés sur la signature S avec les variables de X. Nous distinguerons $T_p(S, X)$, l'ensemble des termes prédicatifs et $T_f(S, X)$, l'ensemble des termes fonctionnels. Dans un univers géométrique, l'ensemble S contient habituellement les sortes correspondant aux types géométriques *point*, *droite*, *cercle*, *longueur* et *angle* et les ensembles F et P contiennent des symboles permettant d'exprimer des constructions d'objets géométriques et de poser des contraintes de distance entre deux points, de distance entre un point et une droite, d'angle entre deux droites, de tangence, d'incidence, *etc.*. Remarquons que si la signature d'un univers géométrique n'est pas toujours explicite dans les logiciels de résolution de contraintes dédiés à la CAO, il est bel et bien présent dans tous ces systèmes ne serait-ce que pour déterminer l'interface utilisateur.

Dans l'exemple 2.1 nous avons utilisé implicitement les sortes `point`, `droite`, `cercle`, `longueur` et `angle`, les symboles prédicatifs `parall` pour exprimer le parallélisme, `est_sur` pour exprimer l'incidence, `tgt` pour exprimer la tangence et enfin `=` pour exprimer des égalités de longueur et d'angle. L'égalité est associée, ici, aux symboles fonctionnels `dist` et `angle` pour exprimer les contraintes de longueur et d'angle. Chaque sorte, symbole prédicatif et symbole fonctionnel possède sa sémantique géométrique usuelle dans le plan euclidien : à la sorte `point` on associe le plan euclidien (assimilé à $\mathbb{R}^2$), à la sorte `droite` on associe l'ensemble des droites du plan euclidien, *etc.* ; au symbole fonctionnel `dist` on associe la fonction de $\mathbb{R}^2 \times \mathbb{R}^2$ dans $\mathbb{R}_+$ qui au couple (A, B) associe $\sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$, *etc.* ; et enfin au symbole prédicatif `parall` on associe la fonction booléenne qui à tout couple de droites associe *vrai* si elles sont parallèles et *faux* sinon, *etc.* Lorsqu'un symbole fonctionnel désigne plus d'un objet, nous l'interprèterons par une *multifonction*. Une manière plus rigoureuse de procéder consiste à définir autant de symboles fonctionnels que d'objets possibles comme cela est fait dans [Sch01].

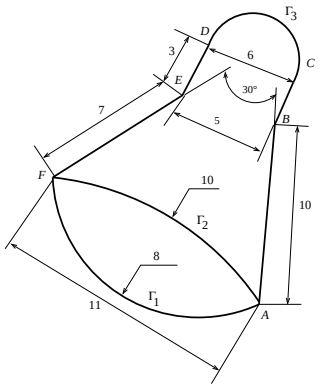
Système de contraintes Habituellement, un système de contraintes est une conjonction de termes prédicatifs. De manière plus précise, nous définissons un système de contraintes comme étant un triplet $\mathcal{S} = (\mathcal{C}, \mathcal{X}, \mathcal{A})$ où $\mathcal{C}$ est une conjonction de termes prédicatifs de $T_p(S, \mathcal{X} \bigcup \mathcal{A})$, $\mathcal{X}$ est l'ensemble des inconnues et $\mathcal{A}$ est l'ensemble des paramètres. On doit évidemment avoir $\mathcal{X} \bigcap \mathcal{A} = \emptyset$.

Lorsque $\mathcal{A}$ est vide, on a un système sans paramètre, *i.e.* un système purement numérique où les seules variables qui interviennent sont les inconnues de $\mathcal{X}$. Mais, il peut arriver parfois qu'à la place de valeurs numériques pour les cotes on ait des symboles. C'est par exemple le cas des catalogues de pièces décrivant une norme. On parle alors de figures paramétrées qui correspondent à un système de contraintes paramétrées.

Exemple 2.1 Dans l'esquisse cotée donnée à la Fig. 1, certaines contraintes sont explicites comme le fait d'imposer que la distance du point A au point F soit égale à 11. D'autres contraintes sont ambiguës : c'est par exemple, la distance entre les points B et E qui est imposée par la double flèche cotée 5 et non la distance entre les droites (BC) et (ED) qui doit être égale, elle, à 6. Enfin, il y a des contraintes implicites comme le parallélisme entre les droites (BC) et (ED). La table à droite de l'esquisse cotée explicite toutes les contraintes imposées en utilisant la signature décrite plus haut. □

Solutions d'un système de contraintes. La notion de solution d'un système de contraintes $\mathcal{S} = (\mathcal{C}, \mathcal{X}, \emptyset)$ sans paramètres *dans* un univers géométrique (*i.e.* en faisant référence à une sémantique précise) est facile à définir. Par exemple, dans le cadre du plan euclidien, on considère les valuations réelles d'un ensemble de symboles $\mathcal{X}$ typés par des sortes géométriques. Rappelons qu'une telle valuation est une fonction $\nu : \mathcal{X} \rightarrow \bigcup_i \mathbb{R}^i$ associant à chaque symbole un n-uplet de réels correspondant aux coordonnées associées. Une valuation peut naturellement être étendue aux termes et aux formules en suivant l'interprétation $\mathcal{I}$ pour donner une interprétation particulière notée $\mathcal{I}_\nu$. Une valuation réelle ν est *solution d'un système de contraintes* $\mathcal{S} = (\mathcal{C}, \mathcal{X}, \emptyset)$ si et seulement si $\mathcal{I}_\nu(\mathcal{C}) = vrai$.

On peut aussi ordonner les inconnues de $\mathcal{X}$ pour donner le n-uplet $(x_1, \ldots, x_n)$. En notant, abusivement



dist(A,F) = 11
dist(A,B) = 10
dist(B,E) = 5
dist(E,D) = 3
dist(F,E) = 7
dist((BC),(ED)) = 6
(BC) // (ED)
angle((AB),(EF)) = 30
rayon(Γ_1) = 8

rayon(Γ_2) = 10
A est_sur Γ_1
A est_sur Γ_2
F est_sur Γ_1
F est_sur Γ_2
tgt(Γ_3, (ED), D)
tgt(Γ_3, (BC), C)

Figure 1: Exemple d'esquisse cotée (à gauche) et contraintes (à droite)

$\nu(\mathcal{X}) = (\nu(x_1), \ldots, \nu(x_n))$, on retrouve la notion classique de *figure solution*. On appelle alors *espace des solutions* l'ensemble des figures solutions. En définissant une *distance* sur cet espace, on peut aussi définir la notion de solution approchée à ε près.

Les définitions précédentes sont celles appliquées par tous les solveurs de contraintes, formels ou non. En revanche, lorsqu'on considère un système paramétré $\mathcal{S} = (\mathcal{C}, \mathcal{X}, \mathcal{A})$, la notion de solution peut être vue à deux niveaux. On peut se placer au *niveau formel* : il faut alors compléter la signature donnée plus haut pour faire de l'univers géométrique une vraie *théorie logique* en ajoutant des axiomes et des règles d'inférence : cette théorie logique est censée avoir l'univers géométrique comme modèle. Une *solution formelle* est alors dans ce cas une valuation formelle $\nu : \mathcal{X} \to T(S, \mathcal{A})$ de sorte qu'en substituant les inconnues de $\mathcal{X}$ à l'aide de ν, $\mathcal{C}$ devienne un théorème de l'univers géométrique. On peut aussi se placer au niveau interprétatif en définissant la notion de valuation des paramètres ν_a comme précédemment. On ordonne alors l'ensemble des paramètres et on appelle *espace des paramètres*, l'ensemble $\mathbb{R}^n$ correspondant aux coordonnées de tous les paramètres : on note V_a cet espace. On étend alors ν_a sur tous les termes de l'univers géométrique pour considérer le système $\nu_a(\mathcal{S}) = (\nu_a(\mathcal{C}), \mathcal{X}, \emptyset)$. De plus, on identifie $\nu_a(\mathcal{A})$ à un m-uplet de valeurs pour les paramètres. On dit alors qu'une fonction de $f : V_a \to V_x$ est une *solution* de $\mathcal{S} = (\mathcal{C}, \mathcal{X}, \mathcal{A})$ si et seulement si pour toute valuation ν_a des paramètres, $\mathcal{X} \to f(\nu_a(\mathcal{A}))$ est une solution de $\nu_a(\mathcal{S})$.

Systèmes bien contraints. On dit, classiquement, qu'un système $\mathcal{S}$ est bien contraint s'il possède un nombre fini non nul de solutions. Cette définition pose un problème lorsque le système est paramétré : il arrive souvent que pour certaines valuations ν_a des paramètres, le système $\nu_a(\mathcal{S})$ soit bien contraint alors que pour d'autres il ne possède soit aucune solution, soit une infinité. Pour un tel système, nous dirons qu'il est bien contraint sur le domaine $\mathcal{D}$ si pour toute valuation des paramétres ν_a à valeur dans $\mathcal{D}$ le système $\nu_a(\mathcal{S})$ est bien contraint. On considérera alors des fonctions solutions au sens ci-dessus dont le domaine de définition sera $\mathcal{D}$. Le sous-ensemble de l'espace des paramètres pour lequel il y a 0 ou une infinité de solutions est appelé ensemble de dégénérescence et le système résultant est dit être dans un cas dégénéré. On dit qu'un système est sous-contraint s'il admet un nombre infini de solutions et sur-contraint s'il n'admet aucune solution.

Solveur. Un solveur est un élément logiciel dont l'objet est de résoudre des systèmes de contraintes ou d'équations. On dit qu'un solveur est correct, vis-à-vis d'une interprétation $\mathcal{J}$, si et seulement si pour tout système de contraintes $\mathcal{S}$, toute valuation des inconnues produite par le solveur est effectivement une solution de $\mathcal{S}$ suivant l'interprétation $\mathcal{J}$. La correction est une propriété éminemment souhaitable pour un solveur, pourtant, on se contente souvent en CAO de la propriété plus faible de correction à ε près. La notion de complétude est également polymorphe : on peut dire qu'un solveur est complet relativement à un système, s'il est capable de fournir *toutes* les solutions de ce système dans le cas où il est bien contraint, ou, mieux, dans tous les cas. On peut également dire qu'un solveur est complet relativement à un ensemble de systèmes de contraintes si il est complet pour chacun de ces systèmes. Par exemple, [VSR92] montre que la méthode de Sunde fournit des solutions pour la classe des polygones semi-simples. Dans le cas particulier où l'ensemble des systèmes envisagés correspond à toutes les instances possibles d'un système paramétré, on parle de *robustesse*. Plus précisément, un solveur est robuste relativement à

un système de contraintes paramétré s'il est capable de fournir toutes les solutions quelles que soient les valeurs des paramètres. Il doit donc, en particulier, être capable de détecter les cas où il y a 0 solution et les cas où il y a une infinité de solutions. Dans ce dernier cas, il devrait être capable de *décrire* l'espace des solutions.

Invariance par déplacement. Une des caractéristiques remarquables des systèmes de contraintes obtenus à partir d'une esquisse cotée en CAO réside dans le fait qu'ils sont, par essence, sous-contraints. Cependant, dans les bons cas, les solutions se déduisent toutes d'un nombre fini d'entre elles en utilisant des isométries directes, qu'on appelle aussi *déplacements*. En effet si une pièce, objet rigide, est correctement décrite, sa position dans le plan n'a aucune importance : on dit qu'on a un nombre fini de solutions *modulo* les déplacements. La plupart du temps, on *fixe un repère* pour déterminer un ensemble fini de solutions pariculières. Ceci peut se traduire dans le système de contraintes initiales par l'ajout de contraintes particulières dites *contraintes de repère*. Un système est alors bien contraint modulo les déplacements si le système initial augmenté de contraintes de repère est bien contraint. Inversement, pour pouvoir retrouver toutes les solutions possibles, il faut introduire la sorte géométrique des déplacements et des fonctions pour les construire à partir des points, droites, *etc.*

3 Rapide état de l'art

Depuis les travaux initiaux de I. Sutherland dans Sketchpad qui utilisait une méthode par relaxation pour résoudre des systèmes de contraintes géométriques, un certain nombre d'équipes se sont penchées sur le problème en suivant des approches diverses. Nous ne citons ici que les principales directions.

3.1 Méthodes analytiques

Dans ces méthodes, on traduit immédiatement les objets géométriques par des réels et les contraintes qui les lient par des équations, algébriques ou non. On emploie ainsi, une sémantique $\mathcal{I}$ traduisant les objets et les contraintes dans un domaine *réel* : soit les décimaux en virgule flottante pour les méthodes numériques, soit une modélisation théorique des réels dans le cas des méthodes formelles. Des méthodes analytiques classiques sont ensuite employées pour résoudre les systèmes d'équations.

Parmi les méthodes numériques employées, on trouve la méthode de Newton-Raphson de convergence rapide mais au comportement chaotique [LM95]. Une méthode alternative consiste en la méthode homotopique, plus lente mais plus sûre. Des travaux de résolution de contraintes utilisant l'arithmétique des intervalles commencent à voir le jour : si ces méthodes semblent lentes, pour l'instant, on peut espérer contrôler la précision des résultats ce qui n'est pas le cas pour les méthodes précédentes. Pour pallier à la fois les problèmes de stabilité et les problèmes de lenteur, certains chercheurs [LM97, HLS01b, HLS01a] ont recours à des méthodes de décomposition combinatoires de systèmes d'équations basées sur le théorème de König-Hall et des algorithmes de flux maximaux.

En ce qui concerne les méthodes formelles, leur complexité en temps les rend impropres aux applications de la CAO où les systèmes d'équations sont de grande taille. Néanmoins, l'implantation de la méthode de Lebesgue [Che92] même si elle est impraticable donne une bonne référence théorique sur les constructions à la règle et au compas.

3.2 Méthodes combinatoires

La sémantique $\mathcal{I}$ utilisée ici traduite les systèmes de contraintes en termes de graphes. Les objets géométriques sont ainsi interprétés par les sommets du graphe, éventuellement étiquetés par le degré de liberté du type de l'objet en question. Les contraintes binaires sont traduites par des arêtes reliant les sommets contraints, éventuellement étiquetées par le degré de restriction correspondant à la contrainte. C'est ainsi que la figure cotée de gauche à la Fig.2 est traduit par le graphe à droite sur la même figure. Notons aussi que cette approche peut-être étendue aux contraintes n-aires pour donner des hypergraphes [HLS01b, HLS01a].

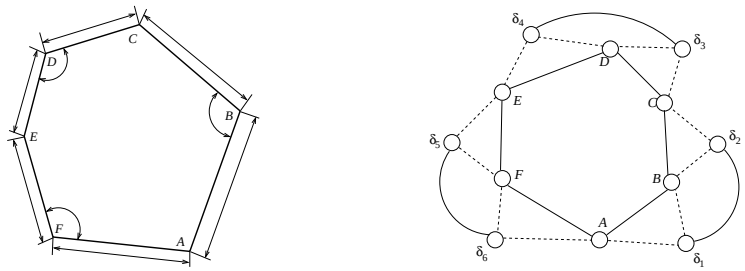


Figure 2: Traduction en graphe

Les méthodes *ascendantes* procèdent alors par propagation de contraintes dans le graphe après avoir fixé deux sommets [BFH+95, AAJM93] : si deux sommets sont *connus* et reliés à un sommet *inconnu* alors ce dernier devient connu. On obtient ainsi une sous-partie rigide du graphe initial qui est nommé *cluster* par Hoffmann. Lorsque cette propagation devient impossible, on recommence ailleurs pour trouver de nouveaux clusters. Lorsque cela est possible, le solveur fait de l'*assemblage* de clusters : cet assemblage est décidé sur des critères combinatoires (voir Fig. 3). Suivant les stratégies de résolution, c'est la propagation de contraintes ou l'assemblage qui est favorisé : c'est ainsi que [FH96] montre la convergence géométrique du solveur de EREPS. Notons que la méthode de Sunde [Sun86, VSR92] peut alors être vue comme une variante de la méthode de Hoffmann avec pour stratégie l'assemblage d'abord, et une variété beaucoup plus grande de méthodes d'assemblage.

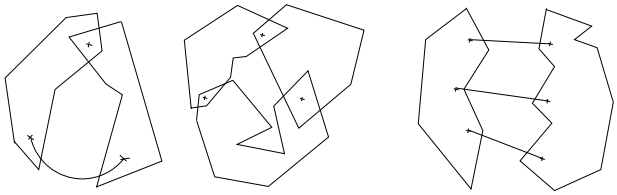


Figure 3: Assemblages de clusters

Inversement, certains chercheurs ont suivi une approche *descendante* consistant à décomposer *a priori* le système de contraintes suivant certains critères. C'est ainsi que Owen cherche les composantes triconnexes et que Sitharam *et al.* recherchent les sous-graphes *denses* minimaux. Dans les deux cas, la résolution effective des composantes touvées et/ou de l'assemblage de ces composantes n'est pas décrite.

3.3 Méthodes géométriques

Dans les méthodes géométriques formelles comme celles décrites dans [Ald88, DMS98], on n'interprète pas la signature, ni le système de containtes : la résolution se déroule, dans une première phase, dans une théorie formelle avec des axiomes spécialisés pour faire des constructions géométriques à la manières de celles enseignées dans les Lycées (voir par exemple [Sch01, Sch94]). Le résultat de cette phase est ainsi une valuation formelle $\nu : \mathcal{X} \to T_f(\Sigma, \mathcal{A})$ qu'on interprète, dans une deuxième phase, comme un *plan de construction* implantant une fonction $f : V_a \to V_x$ solution du système de contraintes. Un des avantages de cette méthode est qu'elle traite naturellement les systèmes paramétrés puisqu'il n'y a aucune évaluation. Cet avantage est accru lorsque les valeurs des cotes sont abstraites au vol pour donner des paramètres d'un système de contraintes générique. Lorsque le plan de construction est interprété avec les valeurs de cotes demandées, on obtient la ou les solutions désirées, mais on peut ensuite facilement faire varier ces valeurs : la construction formelle n'est pas à refaire.

La grande avancée, à notre avis, de l'approche que nous avons développée dans notre équipe [DMS98] consiste en la prise en compte au niveau formel, de l'invariance par déplacement en permettant de fixer et de relâcher des *contraintes de repère*. Cette approche explique aussi la notion de lien virtuel

utilisée par Owen. Ainsi, pour résoudre formellement un système de contraintes géométriques, le solveur *Yams*, issu de nos travaux sur les constructions géométriques, fixe un repère et utilise un agent local, de préférence formel, travaillant sur les contraintes initiales et celles déduites des sous-figures déjà résolues. Lorsque deux sous-figures résolues, définies chacunes par un plan de construction, partagent suffisamment d'éléments géométriques pour définir un repère, le solveur calcule le déplacement qui fait coïncider lesdits éléments et l'applique à l'une des sous-figures résolues. Le plan de construction résultant inclut les plans pour les deux sous-figures, le calcul du déplacement et son application aux objets définis par l'un des sous-plans.

Lorsque tous les objets géométriques cherchés sont définis dans un même plan de construction, la phase de résolution formelle s'arrête avec succès : le plan de construction constitue une solution exacte du système de contraintes. On a ainsi une construction formelle d'un système de contraintes où le type de décomposition indiqué dans la section précédente est bien pris en compte.

On obtient finalement une méthode ascendante avec assemblage, semblable, en un sens, aux méthodes ascendantes évoquées plus haut, mais pour laquelle les solveurs locaux sont aussi puissants que l'on souhaite et l'assemblage réduit à sa vraie dimension de calcul d'un déplacement.

3.4 Discussion

L'objectif sous-jacent dans les méthodes combinatoires, surtout les méthodes descendantes, est –sans doute– de gagner en généralité : on a des procédures générales de décomposition, des critères généraux sur ce qu'est une bonne décomposition et on suppose qu'on a des méthodes générales pour résoudre les petits morceaux et les assembler.

A l'inverse, lorsqu'on utilise des systèmes experts (ou plus généralement des méthodes exactes), la classe des problèmes résolus est forcément limitée. L'attitude des chercheurs dans le domaine a souvent été alors de dire : on peut peut-être rajouter des règles pour savoir résoudre de nouveaux problèmes, mais où s'arrête-t-on ?

Il nous semble, mais c'est un conjecture, que la problématique des constructions géométriques dans un univers assez gros n'est pas décidable. Les méthodes générales sont donc forcément des méthodes approchées. Et les méthodes formelles, exactes, sont donc forcément incomplètes.

Or, nous pensons que c'est un atout de pouvoir résoudre formellement la plus grande partie possible d'un système de contraintes. C'est pourquoi l'approche que nous avons développée, dans l'équipe IGG du LSIIT, privilégie les méthodes formelles lorsque c'est possible, mais utilise aussi des méthodes numériques pour résoudre une partie du problème, la plus petite possible, pour essayer de débloquer les choses et continuer avec des méthodes formelles ensuite [SDM98]. On conserve ainsi la généralité des solveurs analytiques mais en limitant la résolution non-formelle autant que faire se peut.

Le reproche le plus pertinent adressé aux méthodes formelles tient en la relative lenteur du processus de résolution. Le solveur implanté par suit une architecture de système multi-agents, chaque agent implantant un solveur local plus ou moins spécialisé. Le fait d'avoir plusieurs petits solveurs permet de dominer la complexité pour chacun d'eux : en réalité, en considérant un agent implantant la propagation de contraintes simples, on retrouve, suivant les stratégies, la méthode décrite par Hoffmann *et al.* avec son efficacité. Le fait de spécialiser les solveurs locaux permet d'avoir des temps de réponse raisonnables même lorsque la construction géométrique est plus compliquée. Un point de vue semblable était exprimé dans [MM88]. Yams n'est qu'un prototype aux structures de données non optmisées, l'un des travaux actuels consiste en l'étude de la compilation des systèmes à base de règles et son application à Yams.

L'approche formelle que nous avons suivie présente deux gros avantages : proche de la géométrie, elle est relativement indépendante des structures de données et peut s'étendre sans remettre en cause fondamentalement la démarche initiale. Ensuite, le résultat formel peut être exploité de plusieurs manières pour rendre l'interface utilisateur plus conviviale. Nous détaillons ces points dans la section suivante en indiquant les travaux que nous avons déjà effectués et les perspectives possibles.

4 Tirer parti de l'approche formelle

Processus de résolution. L'extension la plus immédiate des capacités de résolution de consiste en l'adjonction de nouveaux solveurs locaux. L'architecture multi-agent se prête bien à ce genre de modularité, mais, pour le moment, l'expert géomètre qui a envie d'étendre doit programmer en C. Il est tout à fait envisageable de décrire précisément un méta-langage de programmation pour les constructions géométriques et de réaliser une interface pour l'expert lui permettant d'étendre sans avoir besoin de programmer. Ce genre de développement, très coûteux en temps de programmation et peu reconnu par la communauté scientifique, est, pour le moment, inenvisageable dans le cadre académique.

Une autre direction de recherche consiste à explorer d'autres géométries. C'est ce qu'a fait l'équipe de P. Macé en proposant le solveur *Desargues* dans le projet *GINA* [LKM98]. Ce solveur travaille dans le cadre de la géométrie projective dans lequel on considère des problèmes comme le suivant, qui nous a été communiqué par D. Michelucci :

Exemple 4.1 On se donne quadrilatère convexe $ABCD$ dans le plan (Fig.4); c'est la perspective d'un carré. On demande de tracer la perspective d'un carré accolé congruent, tel $ABC'D'$. □

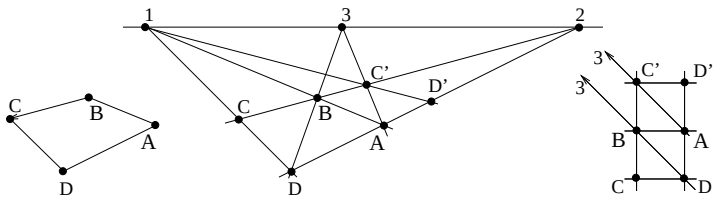
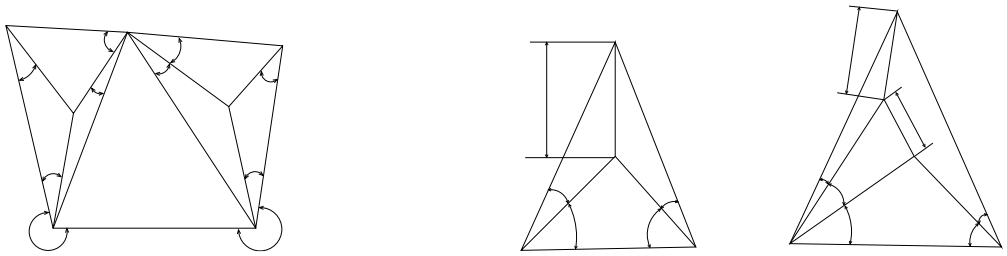


Figure 4: Construction de $ABC'D'$, perspective du carré accolé à $ABCD$.

De manière semblable, et c'est l'un des axes de notre recherche actuelle, on peut considérer le cadre de la géométrie conforme associée à l'invariance par similitudes. L'approche par assemblage décrite plus haut se généralise immédiatement aux problèmes invariants par similitudes, même si les exemples où de la décomposition intervient effectivement sont un peu plus compliqués (voir l'exemple de la Fig.5(a) où toutes les contraintes sont des contraintes d'angle).



(a) avec décomposition

(b) avec libération de distances

Figure 5: Problèmes de construction liés aux similitudes

Notre approche de la résolution est alors celle que nous avons décrite plusieurs fois dans le cas de l'invariance par déplacement : une méthode spécialisée est utilisée pour résoudre tout ce qu'elle peut, un assemblage est éventuellement produit sinon le bord est calculé pour éventuellement servir dans la construction d'une autre sous-figure. La méthode locale employée est un système à base de règles définissant les lieux de points et droites définis par des angles et des rapports de distances comme, par exemple, les règles découlant des propriétés suivantes :

- le théorème de l'arc capable
- le lieu des points M tels que $\frac{AM}{BM} = k$ est un cercle ou une droite

Les problèmes de construction issus de la CAO ne sont généralement pas invariant par similitudes, on peut cependant facilement leur donner cette propriété soit en "oubliant" la contrainte de distance dans le cas où il y en a une (esquisse cotée de gauche de la Fig.5(b)), soit en remplaçant toutes les contraintes de

distances par des contraintes de rapport de distances lorsqu'il y a plus de deux contraintes de distances (esquisse cotée de droite de la Fig.5(b)). La fixation d'un repère pour les similitudes redonne alors toutes les contraintes de distances si le repère choisi est un segment initialement contraint, ou un autre problème (comme dans les deux exemples donnés) dont on espère qu'il est plus simple à résoudre.

Les deux exemples de la Fig.5(b) se résolvent relativement facilement en utilisant un repère adéquat (constitué de la base du triangle dans les deux cas). Dans le deuxième cas, où un rapport de longueur intervient, il est difficile de mettre le problème sous forme de graphe : si on voulait absolument utiliser une structure de données, on se tournerait plutôt vers un hypergraphe. Si les méthodes de décomposition exposées par M. Sitharam semblent continuer de fonctionner, en revanche une méthode locale par propagation de contraintes devient alors plus problématique.

On peut évidemment définir par décomposition toute une classe de problèmes où interviennent libération de contraintes et décomposition de problèmes modulo le groupe des similitudes. Il est aussi facile de concevoir des exemples où une partie du problème se résout par une libération/invariance par similitude et le reste se résout par invariance par déplacement. Notre thème de recherche actuel concernant les solveurs est précisément focalisé sur ce point : comment utiliser aux mieux les divers groupes d'invariance sur une figure pour faire des constructions géométriques formelles.

Pré et post traitement. L'approche formelle apporte également certains avantages au niveau des interactions avec l'utilisateur, depuis les opérations de saisie du problème jusqu'à l'utilisation *a posteriori* de la figure résolue, en passant par divers types d'aide à la résolution de problèmes pouvant intervenir au cours des processus de résolution, d'interprétation et de sélection d'une solution. Voici présentées de façon non exhaustive quelques-unes des possibilités offertes par l'approche que nous avons suivie, ainsi que des dialogues homme-machine qui pourraient apporter une aide à l'utilisateur dans la résolution des problèmes.

Tout d'abord, plaçons-nous dans le cas idéal où la résolution formelle ainsi que l'interprétation numérique ont correctement résolu un système bien contraint. Il y a donc un nombre fini, mais pouvant être considérable, de solutions produites. Dans ce cas, le système apporte une aide précieuse à l'utilisateur, qui sans lui devrait passer en revue toutes les solutions. L'aide se situe à deux niveaux : celui des opérations d'exploration à proprement parler, mais aussi celui de la visualisation.

Plusieurs opérations d'exploration de l'espace des solutions existent déjà dans Yams sous la forme d'un module nommé *Samy*. Celui-ci offre différentes possibilités pour arriver à sélectionner une solution, soit de manière automatique, soit de manière interactive. La première méthode, appelée gel d'une branche, et possible uniquement si les contraintes sont purement métriques, permet de sélectionner automatiquement la solution ressemblant le plus à l'esquisse au sens des déformations continues (voir [EVSD00]). Cette approche pourrait être généralisée à la sélection de la meilleure solution selon divers critères choisis par l'utilisateur selon ses besoins.

La seconde méthode s'inspire des débogueurs que l'on trouve dans la plupart des environnements de programmation, en tirant parti de la structure du programme de construction. Elle permet d'effectuer la phase d'interprétation pas à pas, en marquant une pause dès que l'on rencontre une définition comportant une multifonction induisant un choix. Les différents résultats possibles pour cette multifonction sont alors proposés à l'utilisateur qui en choisit un, guidant ainsi l'interprétation jusqu'à son terme. Cette méthode s'accompagne de diverses possibilités telles que le guidage par gel d'une branche (le premier résultat proposé est celui qu'aurait choisi le gel d'une branche), ou la possibilité de revenir en arrière dans le programme de construction si la voie choisie n'aboutit pas à la solution escomptée.

Une troisième méthode, reposant sur la structure arborescente de l'espace des solutions, permet de manipuler une figure solution à la souris. L'utilisateur sélectionne la partie qui à son sens est mal positionnée, et le prototype lui propose les alternatives possibles pour ces éléments, satisfaisant toujours les contraintes de départ, en se basant sur l'arbre des solutions. L'utilisateur en choisit une, et la figure est mise à jour.

Dans l'utilisation de ces différentes techniques de sélection de solution et d'exploration de l'espace des solutions, la visualisation joue un grand rôle car plus une figure est complexe, plus il devient difficile de sélectionner des objets de cette figure, de repérer les parties devant être sujettes à manipulation, etc. Déjà, dans ces mécanismes, certaines aides de lecture existent, comme la mise en surbrillance des éléments courants à la fois sur l'esquisse et sur la solution. Mais des améliorations restent encore à apporter.

Une des premières améliorations serait d'appliquer un système de couches d'affichage, dans le style des calques utilisés dans beaucoup de logiciels de traitement d'images, qui pourraient sur demande de l'utilisateur être visibles ou non. Par exemple, on pourrait imaginer de superposer à la figure un calque contenant les objets auxiliaires de construction (cercles, droites, etc. non présents sur l'esquisse et créés lors de la phase de résolution pour les besoins de la construction), afin de mieux contrôler les opérations d'exploration de solutions. On pourrait imaginer également des calques différents pour chaque sous-figure, pouvant être affichés avec des couleurs différentes afin de mieux les repérer et de pouvoir ainsi effectuer des opérations sur certaines sous-figures séparément.

Un autre style d'exploration de l'espace des solutions consiste à jouer sur les valeurs de paramètres. Un premier exemple important de cet aspect consiste à tirer parti du plan de construction formel conjointement à une méthode d'arithmétique des intervalles pour gérer les tolérances de cotes. Un deuxième exemple réside dans la possibilité d'animer de manière réaliste des figures contraintes. Ceci est rendu possible grâce à la gestion des sauts d'une branche à l'autre dans l'arbre des solutions lorsqu'on atteint un cas limite.

Plaçons-nous maintenant dans le cas courant : il est bien connu que définir un objet à l'aide de contraintes relève de la programmation [LM97], donc il est fréquent que des problèmes surviennent soit lors de la pose des contraintes, soit lors de la résolution formelle, soit lors de l'interprétation. Ici encore, une approche formelle peut aider l'utilisateur.

En effet, pour qu'une figure soit bien définie, il faut qu'elle soit bien contrainte. Cet objectif n'est pas toujours facile à atteindre pour l'utilisateur pose la cotation. C'est ainsi que sur un exemple compliqué, avec des centaines de contraintes ou plus, il est facile de vérifier que le nombre de contraintes restreint bien tous les degrés de liberté, mais il est en revanche plus difficile, s'il manque une contrainte, de voir où la placer. Inversement, s'il y a trop de contraintes, il est délicat de trouver laquelle est surnuméraire. D. Michelucci a proposé dans [LM97] une méthode à base de flux maximaux pour déterminer les parties bien-, sur-, et sous-contraintes. Cependant, à la manière de Sunde [Sun86], on peut imaginer dans une méthode incrémentale de construction, à base géométrique, permettant de déterminer les parties rigides au fur et à mesure que l'utilisateur entre les contraintes. De plus, en cas de conflit, cette méthode présenterait l'avantage de proposer des explications de nature géométrique sur le problème rencontré.

En théorie, la résolution formelle ne peut jamais échouer. Dans le pire des cas, un ou plusieurs sous-systèmes de contraintes auront été traduits en systèmes d'équations résolus numériquement pendant la phase d'interprétation. Dans ce cas-là, l'utilisateur peut souhaiter voir quelles sont les parties résolues numériquement, et décider de faire la construction lui-même. Il serait donc intéressant d'offrir à l'utilisateur cette possibilité, en quelque sorte d'ajouter un agent de résolution "humain". Ceci pourrait être associé à un module d'apprentissage par l'exemple.

5 Conclusion

Malgré la part relativement restreinte qu'occupe l'approche géométrique et formelle dans le domaine des contraintes géométriques de type CAO, il nous semble qu'elle possède de nombreux atouts. Tout d'abord, elle a le mérite de poser très précisément le problème. Ensuite la réutilisabilité des plans de construction permet réellement de manipuler un objet défini par un système de cotes : de simuler des variations de cotes, de regarder ce qui se passe lorsqu'on met des tolérances sur les cotes, de faire de l'animation de figures et enfin, d'explorer relativement facilement l'espace des solutions.

De plus, les inconvénients souvent relevés à l'encontre des constructions formelles nous paraissent exagérés. Ainsi, à condition d'accepter que le solveur formel traite certaines parties du problèmes avec une méthode approchée, on n'a pas de perte de généralité par rapport à un solveur analytique et on conserve les avantages de l'approche formelle pour le reste de la figure. Par ailleurs, la lenteur, souvent reprochée aux systèmes à base de règles pourrait être relativisée en utilisant un système multi-agent, éventuellement couplé avec des processus légers, et des structures de données adéquates.

Finalement, nous montrons dans ce papier, que notre approche offre une grande souplesse dans le choix des méthodes de résolution. En outre, nous donnons des exemples de géométries non euclidenne où elle peut être étendue. Notre travail actuel concerne notamment la géométrie conforme, mais la géométrie

projective, utilisée dans d'autres équipes, est également envisageable.

References

[AAJM93] S. Ait-Aoudia, R. Jegou, and D. Michelucci. Reduction of constraint systems. In *Compugraphic*, pages 83–92, Alvor, Portugal, 1993. Also available at *http://www.emse.fr/~micheluc/*.

[Ald88] B. Aldefeld. Variations of geometries based on a geometric-reasoning method. *Computer-Aided Design*, 20(3):117–126, 1988.

[BFH+95] W. Bouma, I. Fudos, C. Hoffmann, J. Cai, and R. Paige. Geometric constraint solver. *Computer-Aided Design*, 27(6):487–501, 1995.

[Che92] G. Chen. Les constructions à la règle et au compas par une méthode algébrique. Technical Report Rapport de DEA, Université Louis Pasteur, 1992.

[DMS98] J.-F. Dufourd, P. Mathis, and P. Schreck. Geometric construction by assembling solved subfigures. *Artificial Intelligence Journal*, 99(1):73–119, 1998.

[EVSD00] C. Essert-Villard, P. Schreck, and F. Dufourd, J. Sketch-based pruning of a solution space within a formal geometric constraint solver. *Artificial Intelligence*, 124:139–159, 2000.

[FH96] I. Fudos and C. M. Hoffmann. Correctness proof of a geometric constraint solver. *International Journal of Computational Geometry and Applications*, pages 405–420, 1996.

[HLS01a] C. Hoffmann, A. Lomonosov, and M. Sitharam. Decomposition plans for feometric constraint problems, part ii : New algorithms. *J. Symbolic Computation*, 31:409–427, 2001.

[HLS01b] C. Hoffmann, A. Lomonosov, and M. Sitharam. Decomposition plans for feometric constraint systems, part i : Performance measures for cad. *J. Symbolic Computation*, 31:367–408, 2001.

[LKM98] O. Lhomme, P. Kuzo, and P Macé. Desargues: A constraint-based system for 3d projective geometry. *Geometric Constraint Solving and Applications*, pages 196–210, 1998.

[LM95] H. Lamure and D. Michelucci. Solving constraints by homotopy. In *Proc. of the Symp. on Solid Modeling Foundations and CAD/CAM Applications*, pages 263–269, May 1995.

[LM97] H. Lamure and D. Michelucci. Qualitative study of geometric constraints. In B. Brüderlin and D. Roller Editors, editors, *Proceedings of Workshop on Geometric Constraint Solving and Applications, Technical University of Ilmenau, Germany*, pages 134–145, 1997.

[MM88] D. Martin and P. Martin. An expert system for polyhedra modelling. In *Proceedings of the Eurographics Conference*, pages 221–232. North-Holland, 1988.

[Sch94] P. Schreck. Implantation d'un système à base de connaissances pour les constructions géométriques. *Revue d'Intelligence Artificielle*, 8(3):223–247, 1994.

[Sch01] P. Schreck. Robustness in cad geometric construction. In *Proceedings of the fifth International Conference on Information Visualisation, IV2001*, pages 111–116. IEEE, 2001.

[SDM98] P. Schreck, J.F. Dufourd, and P. Mathis. Using a numerical tool in a formal construction method with decomposition. In *2th International Conference on Computer Graphics and Artificial Intelligence*, May 1998.

[Sun86] G. Sunde. A CAD system with declarative specification of shape. In *Proceedings of the IFIP WG 5.2 on Geometric Modeling,* Rensselaerville, 1986.

[VSR92] A. Verroust, F. Schonek, and D. Roller. Oriented method for parametrized computer-aided design. *Computer-Aided Design*, 24(10):531–535, 1992.