CCL: Expressions of Choice in Agent Communication

Steven Willmott, Monique Calisti, Boi Faltings, Santiago Macho-Gonzalez,
Omar Belakhdar, Marc Torrens.

Laboratoire d'Intelligence Artificielle
Ecole Polytechnique Federale de Lausanne
IN (Ecublens), CH-1015 Lausanne, Switzerland.
{willmott, calisti, faltings, santiago, belakdar,
torrens}@lia.di.epfl.ch

Abstract

When shall I fly to New York? Which airline should I choose? How are these related to which airport I arrive at, to how I might travel into the city and to where I choose to stay? Many current and potential applications of agents involve reasoning and communicating about multiple interrelated choices.

To date however, most proposals for communication in agent systems have provided little or no direct support for the type of communication required by these applications. To address this need, this paper describes the Constraint Choice Language (CCL) - an agent Content Language designed to support agent problem solving by providing explicit representations of choices and choice problems.

1. Introduction

Solving problems implies defining and making *choices*. When a group of people decide on a meeting, they agree on choices for the meeting time and place. Resource allocation involves choosing which resources to reserve for which tasks. In planning, we exchange information about task timing and resources as well as commitments on what gets done by whom and when. Many of these tasks lend themselves perfectly to implementation by agent systems.

There are numerous examples of multi-agent systems that solve choice problems from airport slot allocation [6], network management [4, 12] and negotiation about mutually acceptable choices ([11] for example) to numerous distributed meeting scheduling systems. In this work, it is generally assumed that the choices to be made as well as the space of possibilities are *fixed* and agreed on by all agents before problem solving begins. This requirement is difficult to satisfy when problem-solving takes place in a het-

erogeneous environment such as the Internet. Choices are often not known beforehand, but only discovered from information services. For example, when planning a trip it may not a-priori be known what means of transport will be used, how many stops it will involve, and which route it will take. Such information is only gathered incrementally from *information agents*, such as airline, railway, or tourist information services. Since choices are now no longer fixed, negotiation about alternatives requires that agents communicate the choices being considered - generating a growing need for flexible communication about choices.

To date however, most proposals for communication in agent systems have used very general semantic paradigms such as first-order logic. These general paradigms pose serious implementation problems due the intractability of logical inference. In order to build practical agent systems, it is often necessary to adopt a simpler paradigm which is sufficient for most purposes while at the same time being tractable.

To address this need, this paper describes the Constraint Choice Language (CCL) which is an agent *content language*¹ designed to explicitly support agent communication about choices and choice problems. CCL:

- Uses Constraint Satisfaction Problems (CSPs, [1]) as an underlying model for choice problems,
- Addresses a large class of choice problems by providing explicit support for notions such as choices, options for choices and relationships between choices,
- Supports communication about choice problems from problem modelling right through to problem solving,
- Is designed to be used with the FIPA standard Agent Communication Language FIPA-ACL and has been in-

¹See Section 2 for the distinction between agent communication languages and content languages.

corporated in the FIPA 1999 standard as content language *FIPA-CCL*.

This paper aims to provide a brief outline of the ideas behind CCL, details of the language can be found in the complete language specification [16]².

2 Language Description

Communication between agents is often separated into two levels - the *communication language* level and the *content language* level. Expressions in the former describe an agent's mental attitudes towards states of the world and are characterised as *communicative acts*. Expressions in the later describe states of the world and represent the *objects of the communicative acts*. CCL is designed as a content language for states of the world involving notions of choice including:

- Sets of choices to be made,
- Relationships between choices,
- Operations that can be performed on choice problems,
- Propositional statements about choice problems.

As mentioned in the introduction, the language is based on the CSP formalism. In recent years, CSPs have emerged as the most successful paradigm for solving combinatorial choice problems, including many applications in resource allocation, scheduling, diagnosis, spatial reasoning, planning and product configuration. This strong interest has resulted in a wide variety of well-developed techniques, including optimisation and tradeoff analysis. Researchers have also applied CSP to problem-solving in distributed contexts. Work by Yokoo [17] and others has addressed the problem of *solving* CSP in a distributed manner. CSPs have also been used to build lightweight agents for tasks from concurrent engineering [9] to e-commerce [13].

2.1 Definition of a CSP

A finite binary discrete CSP is defined as:

- A finite set of variables V ,
- A finite domain D_i of possible discrete values for each variable $v_i \in V$,
- A finite set of constraints C between any pairs of variables in V .

A solution to the CSP is defined as:

²This is the current version (v2.01) included in the FIPA99 standard. Available on-line at <http://liawww.epfl.ch/CCL/>

An assignment of values to variables in V such that: each variable $v_i \in V$ is assigned a value $d \in D_i$, and none of the constraints $c \in C$ are violated.

Finding a solution therefore involves identifying consistent assignments of values to each variable such that all the constraints posted for the problem are respected. More formal definitions of CSPs can be found in [7] and [1] amongst others. The basic definition has previously been extended in many ways to introduce aspects such as new variable and constraint types or dynamic properties. Each of these CSP sub domains generally has a well defined extension to the basic CSP model and its own specific literature on how to model and treat such problems.

2.2 Expressing Choices and Choice Problems in CCL

In CCL, a choice problem is defined as a CSP in the following way:

- Variables are choices to be made - such as which brand of shampoo to use or how many roses to buy for a date. The set of variables V is the set of choices which all need to be made to have a complete solution to the current problem,
- Domains are the available options for each choice (variable). Thus the number of roses may be anywhere between 1 and 30 and the brands of shampoo one of X, Y and Z,
- Finally, constraints are relationships between choices which express valid or invalid combinations (the number of roses for a date, for example, is likely to be heavily dependent on who the date is with and the current state of the users finances). The set of constraints C therefore restricts the set of all possible combinations of choices made to a smaller set of desirable assignments which meet the requirements of a solution to the choice problem.

The assignment of one of the values from a domain D_i to a variable v_i corresponds to making a choice for v_i . The set of all possible combinations of assignments of domain values to variables defines the problem search space.

2.3 Core Language Constructs

Since CCL is designed primarily to be used with FIPA-ACL [3] (although adaptation for use with other ACLs should be straightforward), the main constructs in the language are classified into three groups: objects, actions and

propositions. These correspond to the three classes of construct specified in [3] for use with FIPA ACL. The following is a complete list of the principle constructs available in CCL:

- **Objects:** The objects in CCL are the following:
 - CSP: represents a complete choice problem and is composed of elements representing the notions of *Variables*, *Domains*, *Constraints* described in Section 2.2,
 - CSP-solution, CSP-solution-list: represent choices made which satisfy all constraints and are composed of *Variable assignments*.
- **Actions:** The available actions in the language are:
 - CSP-give-constraints, CSP-give-values: are information gathering actions adding either more options to choices (give-values) or restricting available combinations of choices (give-constraints),
 - CSP-solve, CSP-solve-list: are choice making actions generating combinations of choices made which respect the information specified for the choice problem (constraints, available values, etc.).

Each of the actions represents a function carried out by an actor (agent) with one of the objects of the language as an argument.

- **Propositions:** The available propositions in the language are:
 - CSP-insoluble, CSP-soluble: which make statements about the existence of solutions to the problem,
 - CSP-unknown: which is an all purpose proposition for cases when a reference failed in the conversation.

The truth/falsehood of propositions, definition of solutions, well formedness of choice problems and other properties such as solution search terminations are all well defined. These definitions are given in terms of the standard formal version of the CSP model defined in Section 2.1 and its interpretation as a choice problem outlined in Section 2.2.³ The number of constructs in CCL is clearly quite small, however each construct may be structured and the flexibility of the language rests upon the flexibility of the underlying CSP/choice problem model.

³See [16] for complete definitions.

3 Using CCL

CCL is primarily intended for expressing states of the world involving interrelated choices. Tasks involving this kind of state can often be broken down into four main steps: *problem modelling*, *information gathering*, *information fusion*, and *problem solving*. This section gives an overview of how CCL can be used in each of these steps.

3.1 Travel Planning Example

To give a concrete feel to the explanations in this section, the following example scenario will be used for illustration:

Caroline would like to meet Liz in London for one of the exhibition preview receptions at the Tate Gallery. These will be held at the beginning of October. Both Liz and Caroline have other appointments around that time, and will need to travel to London from their homes in Paris and New York.

Each of the meeting participants has a Personal Travel Assistant Agent (PTA) and a Diary Agent (DA). The scenario also includes an Air Travel Agent (ATA) able to give information about flight availability, a Hotel Broker Agent (HBA) able to give information on hotels in London and a Constraint Solver Agent (CSA) able to solve generic CSPs.

Caroline's PTA agent takes charge of planning the trip - even in this simple example there are several choices to be made:

- Exact timing of the meeting,
- Which hotel to stay in,
- Flights for both participants.

The PTA's task now involves formalising the problem with initial values and constraints, gathering information for problem solving (from the DA, ATA and HBA agents), composing this information and finally solving the complete problem (with the help of the CSA).

Note that throughout, example messages use a condensed syntax - both for ACL and for CCL (CCL uses an XML syntax) to emphasise the key points in the exchange and reduce required space.

3.2 Problem Modelling

As described in Section 2.2, modelling a choice problem in CCL requires the problem to be formulated as a CSP - that is:

- Identifying the choices to be made - these become the variables in the problem formulation,

- Identifying which options are available for each of the choices - this generates the domains of values for each of the variables,
- Specifying how choices are related - generating the constraints (relations and exclusions) which apply to problem solutions.

Figure 1. A problem modelled as a CSP can be represented as a graph (known as a constraint graph) with variables as vertices and constraints as arcs. This type of visualisation can greatly ease problem specification.

As noted in Section 2, CSPs have established themselves as a powerful formalism for expressing problems involving multiple interrelated choices. Although some experience is required with the framework, the process is in general intuitive (see the graphical representation in Figure 1) and most importantly generates problem descriptions with well defined properties and well studied solution techniques.

Travel Planning Example The travel problem can be modelled as a CSP with variables such as: flight departure times, airports, diary dates, hotels, airlines and meeting places. Constraints which might be known at this initial stage include: travel time between home and airport, duration of the meeting (a constraint between the meeting start and end times). Many of the variables would also have domains which indicate the range of values which are initially acceptable (the days for the meeting might be given as any day in a two week period for example).

3.3 Information Gathering

Once a choice problem had been modelled as a CSP, problem information can be added to the CSP representation to constrain or expand the range of options available (Figure 2). Information can be obtained from other agents by requesting them to:

- Add values to domains using the CSP-give-values action,

- Add constraints over the variables using the CSP-give-constraints action.

These actions can be applied to either the complete CSP or to parts of it. Adding information generated by these actions to a CSP results in a changed solution space (depending on the type of composition applied - see Section 3.4) - more possible options for choices (CSP-give-values) or greater restrictions on possible solutions (CSP-give-constraints).

Figure 2. The agent with the choice problem (agent A) requests information related to the problem from agents with expertise relating to each sub-part of the problem. To achieve this, the CSP is split up into pieces and additional options (values) or constraints are requested for each sub-part.

A further mode of interaction (not shown in the figure) is to send the complete CSP to several agents and asking for constraints or values. This case would be most useful if the agents being queried have similar domains of expertise (e.g. each is an airline booking agent but for a different airline). The agent trying to solve the choice problem would then receive several sets of information for the same sub-parts of the problem and can subsequently choose which offering to use.

The main advantages of CCL for information gathering are:

- The representation allows simple decompositions of the problem being addressed,
- The information that can be added by each of the information agents is well defined,
- The problem and eventual response representations are very compact: a list of options for each choice rather than an *enumerated list of all possible combinations*. The length of an enumerated list is *multiplicative* in the size of the variable domains whereas for a CSP representation the size is *additive* w.r.t the variable domains.

Information agents may also be able to advertise which types of choices they can provide information for by giving CSPs as templates.

Travel Planning Example This phase of the task involves the PTA agent sending information requests to the four agents which have information about the problem: ATA (asking for possible values for the flight variables), the two DA agents (asking for constraints on the meeting times/dates) and the HBA for hotel information. Figure 3 shows an example message from the agent trying to solve the travel problem to the ATA agent.

```
(request
:sender "Personal Travel Agent"
:receiver "Air Travel Agent"
:content
(Action: CSP-give-values
(Object: CSP
:CSP-ref "my-travel-problem"
:variables {
(Element: CSP-variable
:name outward-flight-date
:type time-date
:role {flight-date, outward}
:domain {9-10-00,10-10-00, ...} )
(Element: CSP-variable
:name airline)
(Element: CSP-variable
:name departure-airport)
...
}
:relations { }
:exclusions { }
)))
```

Figure 3. Using a FIPA request message, the PTA requests an action to add values to a problem description (a subset of the PTA's whole problem). Any initial values in variable domains serve as guidelines for information the ATA might add in the rest of the CSP.

3.4 Information Fusion

Having been requested to add information to the CSP, the information agents return new CSPs which contain the information related to the variables and constraints in the CSP sent in the query. The querying agent needs to fuse these to combine the information from the various sources. There are two main ways of combining CSPs: conjunctive combination (s.t. the new CSP satisfies *all* the participant CSPs) and disjunctive combination (s.t. the new CSP satisfies *at least one* of the participant CSPs).

Again the underlying CSP framework is crucial here - just as decomposition of the whole problem into parts was simple and well defined, the generic structure makes it possible to precisely define the operations which allow the re-composition of the information from two or more messages.

For advanced treatment, CSPs can also be transformed into canonical forms such as the microstructure [5, 15] where fusion and equivalence can be decided by efficient algorithms.

Travel Planning Example The responses of the information agents (assuming they were able to perform the requested actions) would be inform messages containing the same CSP sub-problem⁴ that was sent in the request. The CSP may have additional values (in the case of the airline agent for instance) or constraints (in the case of the diary agents for example). These additions represent the information which must be taken into account when looking for a solution.

3.5 Problem Solving

Once a problem has been modelled, information gathered and composed to form a single choice problem, this can be solved. The final choice problem, with all values and constraints gathered, represents a well defined search and solution space. Every valid solution in the space is an acceptable combination of choices the agent could make which 1) represents its original problem specification, 2) represents the constraints and options provided by the information sources.

Figure 4. The problem can be solved as a generic CSP problem by sending it to a third party solving agent which can apply one of a suite of solving techniques.

In the context of the CCL language there are two main ways to solve a constructed CSP problem:

- Implementation of one or several solution algorithms in the agent solving the choice problem (PTA agent A in Figure 4). Solution algorithms range from very simple compact approaches to elaborate specialised techniques. Example algorithms can be found in [14], among others,
- Usage of a third party constraint solver agent which implements a suite of algorithms for solving generic

⁴Note that here the CSP is treated as a propositional object as if it were wrapped in a second order function equivalent to FIPA-SL's (*iota* X (*result* (*act* *i*) X)).

CSPs. Such solver agents can be requested to solve choice problems using the CCL language actions `CSP-solve` and `CSP-solve-list` (see Figure 4).

The semantic meaning behind the variables and constraints in the problem can be stripped away during the solution process and the problem can be solved as a generic CSP (such as the one defined in Section 2.1). This is an important feature of CCL since it allows powerful CSP problem solving algorithms to be applied (in either of the solution methods given above). It also allows problems to be solved to be completely anonymised: names, places, constraints and values can all be removed and replaced with arbitrary tags. This transformation makes it very difficult for a problem solving agent or anybody intercepting the message containing the problem to deduce what the real problem is about.

Travel Planning Example Having sent the completed CSP containing all the gathered information to a CSA agent requesting a `CSP-solve` action the PTA should receive a reply such as the one shown in Figure 5 or a `CSP-insoluble` proposition if there was no available valid combination of choices. If a CSP is insoluble, partial or MAX CSP techniques can be applied to find approximate solutions.

```
(inform
:sender "Constraint Solver Agent"
:receiver "Personal Travel Agent"
:content
  (Object: CSP-solution
   :CSP-ref "my-travel-problem"
   :assignments {
     (Element: CSP-variable-assignment
      :name outward-flight-date
      :value "13-10-00")
     (Element: CSP-variable-assignment
      :name return-flight-date
      :value "16-10-00")
     (Element: CSP-variable-assignment
      :name hotel-name
      :value "Hilton")
     ...
   }
  ))
```

Figure 5. The solution returned is a list of variable assignments - giving a decision for each choice to make which in combination make up a valid solution to the choice problem.

4 Language Demonstrator

To validate the language specification, demonstrate its utility and to provide feedback to the specification process a demonstration application is being developed in parallel with CCL. The application implements a travel planning scenario similar to the one used as an example throughout the previous section and includes:

- PTA and DA agents for each user,
- An ATA agent providing flight information from a database containing a snapshot of a real Swissair flight database.
- A CSA agent for solving CSPs which uses the Java Constraint Library (a freely available library implementing standard CSP algorithms⁵).

All agents were implemented on top of the FIPA'97 compliant Open Source Platform FIPA-OS (see <http://www.nortelnetworks.com/fipa-os/> for more details).

Figure 6. The choices involved in Caroline's attendance at the meeting with Liz are represented by 7 variables. 10 constraints model the relationships between the variables and how they influence the meeting start and end times.

⁵This library can be downloaded at <http://liawww.epfl.ch/~torrens/JCL/>.

A simplified version of the CSP model built for demonstration scenario is shown in Figure 6. Most of the variables are single time/date tokens, the flight variables are flight numbers and the free diary slots (FDS) variable has a domain containing tuples of the form $(start, end)$ where *start* and *end* are time/date tokens signifying the beginning and end of a free period of time in Caroline’s diary. The constraints between FDS, outward flight departure (OFD) and return flight arrival (RFA) are worth noting:

- OFD is constrained w.r.t. the first index of the tuples in FDS with a “greater than” constraint. This expresses that Caroline’s flight must leave after the *start* of the selected free slot in her diary,
- RFA is constrained w.r.t. the second index of the tuples in FDS with a “less than” constraint. This expresses that Caroline’s return flight must arrive before the *end* of the selected free slot in her diary.

Since the variable FDS may only take one $(start, end)$ tuple value in any given solution, these constraints ensure that Caroline completes her entire trip in a single continuous free period. The constraints imposed by Caroline’s schedule affect the possible choices of flights (flight, departure and arrival times) which in turn affect Caroline’s availability for the meeting.

The sub problem of the CSP which is made up of the variables denoted by “C:” in Figure 6 contains all the choices which must be made by Caroline. More meeting participants with similar models can be added by allowing each to express constraints on the meeting start and end times. Each participant may have a different set of variables and constraints in their own subproblem which influence his or her constraints on the meeting times (different modes of transport, dependencies on other meetings etc.).

Once information has been gathered about the variables, domain values and constraints of each meeting participant and from related services such as the air travel agent ATA, the complete CSP can be solved as a single problem.

5 CCL and Existing Content Languages

Currently available agent content languages, such as KIF⁶ ([8], often used with KQML [2]) or FIPA SL (Semantic Language - [3]) are based on very general semantic paradigms such as first order logic. Although powerful, these languages lack standardised ways of expressing choices. While it is of course possible to express a choice

⁶Note that we treat KIF as a content language here although its primary purpose is to enable knowledge description and it does not, for example, allow the representation of actions.

in a predicate calculus expression, the absence of a standard convention means that different agents can formulate choices in different ways. Fusing or comparing choices would then require theorem proving (potentially undecidable). A further difficulty is that since all expressions are built up from a small core of primitives, repeating expressions of structures such as lists of choices, options etc. adds significantly to message length.⁷ Thirdly using a language such as SL or KIF requires developers to understand of its complex logical foundations. Expressions in CCL are arguably much more intuitive since they are defined in the same terms as the problem being addressed (choices to be made, options available and relationships between choices). The fact that CCL only allows binary constraints can, however, complicate problem modelling.

On the minus side, since CCL is less expressive than either SL or KIF, its application domain is also more restricted. As noted above however this restriction can also be seen as positive - expressions in the language are no longer undecidable and the language is much more effective for its target group of applications.

Despite the existence of SL and KIF, the vast majority of agent applications continue to use application and domain specific content languages developed specifically for the project at hand. This approach is clearly often necessary if no suitable standard content languages are available. The advantages of using a generic language such as CCL instead of a home-grown language are potentially two-fold:

1. The use of a common problem model provides access to common resources of experience, research and problem solving tools,
2. CCL’s inclusion in the FIPA standard very much increases the likelihood of future cross application inter operation.

Application in a particular domain would then be done 1) by the problem descriptions used (i.e. the CSP model) and 2) by the domain ontology applied.

Finally, the most relevant work done on communicating choices which does not fall into either of the areas discussed above is that done in the KRAFT project [10]. This work is extensive and applies a form of KQML and a constraint representation to transform and fuse knowledge from multiple database sources.

6 Conclusions

From booking hotels, flights and car hire to purchasing the correctly sized wardrobe and scheduling a management

⁷In fact, representing sets of choices which *have yet to made* is difficult in a logical language based on evaluating the truth/falsehood of a statement.

meeting - many important current and proposed agent applications require reasoning and communication about complex interrelated choices. To date, no available content language has been able to address the communications needs of these applications adequately.

CCL provides a standard way of expressing choices which is modelled after the successful formalism of *constraint satisfaction*. The language represents a significant departure from previously defined generic content languages by being *problem solving focused* - using a common problem representation to underpin all phases of communication. Crucially, CCL also *reduces the complexity of reasoning* required of agents by limiting its expressiveness to the CSP formalism. While this makes it impossible to express certain situations the considerable success of CSP in practical applications shows that this is not a serious handicap.

Although CCL in its present form can already cover a large number of applications, we hope that the language will evolve with usage (adding support for different types of problem and perhaps integration with other languages depending on user needs).

CCL Language Resources: CCL's utility can of course only be proven with usage, hence we are making every effort to support its usage by interested parties so we can eventually improve and refine the language.⁸ To this end, a web page with up to date CCL related resources can be found at <http://liawww.epfl.ch/CCL/>.

Acknowledgements: Comments from members of FIPA's Technical Committee C (Agent Communication) have been an invaluable source of input to the language. Particular thanks go to Mark Greaves, Tim Finin and Donald Steiner for their encouragement and to Nortel Networks for early access to the FIPA-OS agent platform.

References

- [1] R. Dechter. Constraint Networks. In *Encyclopedia of Artificial Intelligence*, pages 276–285. Wiley, 1992.
- [2] T. Finin et al. Specification of the KQML Agent-Communication Language – plus example agent policies and architectures, 1993.
- [3] F. for Intelligent Physical Agents (FIPA). FIPA Agent specification 1997. Technical report, FIPA, homepage - <http://www.fipa.org/>, October 1997.
- [4] A. L. G. Hayzelden, J. Bigham, and Z. Luo. Multi-Agent Interactions for a Network Management System. In A. L. G. Hayzelden and Bigham, editors, *Software Agents for Future Communication Systems*, pages 160–182. Springer, 1999.
- [5] P. Jégou. Decomposition of domains based on the micro-structure of Finite Constraint-Satisfaction Problems. In *Proceedings of AAAI93*, pages 731–736. 1993.
- [6] A. Lucas, M. Ljungberg, R. Evertsz, G. Tidhar, R. Goldie, and P. Maisano. New techniques for air traffic management for single and multiple airports. Technical Report 51, Australian Artificial Intelligence Institute, Melbourne, Australia, 1994.
- [7] A. Mackworth. Consistency in networks of constraints. *Artificial Intelligence*, 8, 1977.
- [8] X. A. H. G. on KIF. Knowledge Interchange Format Specification. Technical report, ANSI, 1995.
- [9] P. Petrie. Agent-Based Engineering, the Web, and Intelligence. *IEEE Expert*, 11(6):24–29, December 1996.
- [10] A. Preece, K. Hui, A. Gray, P. Marti, T. Bench-Capon, D. Jones, and Z. Cui. The KRAFT Architecture for Knowledge Fusion and Transformation. In *To appear: Proceedings Expert Systems 99*. 1999.
- [11] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*. MIT Press, 1994.
- [12] F. Somers. HYBRID: Intelligent Agents for Distributed ATM Network Management. In *Working Notes of the Workshop on Intelligent Agents for Telecom Applications - ECAI'96*. 1996.
- [13] R. Sun, B.-T. Chu, R. Wilhelm, and J. Yao. A csp-based model for integrated supply chains. In *Working Notes of the AAAI'99 Workshop: Artificial Intelligence for Electronic Commerce*, Orlando, Florida, July 1999.
- [14] E. Tsang. *Foundations of Constraint Satisfaction*. Academic Press, 1994.
- [15] R. Weigel and C. Bliet. On Reformulation of Constraint Satisfaction Problems. In *Proceedings of ECAI98*, pages 254–258. 1998.
- [16] S. Willmott, B. Faltings, M. Calisti, S. Macho-Gonzales, O. Belahdar, and M. Torrens. Constraint Choice Language (CCL), Language Specification V2.01. Technical Report 99/320, Artificial Intelligence Lab, Computer Science Department, Swiss Federal Institute of Technology at Lausanne (EPFL), November 1999.
- [17] M. Yokoo. Asynchronous Weak-commitment Search for Solving Large-Scale Distributed Constraint Satisfaction Problems. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 467–473. MIT Press, 1995.

⁸Note that there are no intellectual property issues related to the use of CCL - it is free for non-commercial and commercial use, all we ask is for feedback to help improve the language for others in the future.