VISUALISATION AND INFORMATION

Full Paper

LOGIC PROGRAMMING IN A CONSTRUCTION PLANNING WORKBENCH

**Gerardo Trinidad, Fanny Boulaire,
Cheryl McNamara and Robin Drogemuller**
CSIRO Manufacturing and Infrastructure Technology
Gerardo.Trinidad@csiro.au

ABSTRACT

This paper describes a logic programming approach to construction planning and scheduling. It describes the use of Prolog in generating feasible job logic from an IFC database. The resulting job logic is the basis for generating a first-draft construction schedule. This is one of the core processes in the Construction Planning Workbench project within the CRC for Construction Innovation, which aims to transform a 3D model (e.g. CAD model of a building) into a 4D model (e.g. construction schedule).

This paper also shows that IFC data is a potential source of information in the generation of draft construction schedules. The paper illustrates the feasibility of using logic programming to codify knowledge and trade practices in the construction industry. The use of logic programming, instead of other software development paradigms, allows the development of a flexible and expandable domain rule base.

The results described in this paper are potentially useful in the development of practical and effective tools for generating draft construction schedules from 3D CAD models. This could be beneficial in early planning stages and as a training tool for novice planners.

Keywords: 3D CAD, 4D CAD, Construction Schedule, Logic Programming

1. INTRODUCTION

Construction planning is a fundamental and challenging activity in the management and execution of construction projects. It involves the definition of work tasks, the estimation of required resources and durations of individual tasks, and the identification of any interactions among the different work tasks. A good construction plan is the basis for developing the budget and the schedule for work. The need for planning and schedule control becomes dramatically important with the increasing complexity of a construction project.

Unfortunately, planning and scheduling of construction activities is a time consuming and error prone process where many factors need to be considered simultaneously. In the early stages of a project several plans may be developed to assess alternatives in sequencing, timing and the use of resources. During construction, the higher level plans need to be defined at greater levels of detail within the constraints imposed by the initial planning process and the evolution of the process itself. Consequently, improvements in the support for and speed of the planning of construction processes would improve the efficiency of the industry.

Previous works have tried to integrate the building product model with the construction process model. They presented three important concepts: building product component, construction process component, and the association between the product component and process component (Fischer and Froese, 1996). A possible application of this association between building product and construction process is to automatically generate a construction schedule using some sort of knowledge-based reasoning engine (Chevallier and Russell, 2001).

The goal of this paper is to demonstrate how a logic programming language, such as Prolog, with a rule base, in order to generate draft schedules automatically. The rules serve to capture domain knowledge such as basic construction principles and standard practices with the industry.

1.1 USE OF PROLOG

Forming a construction plan is a backward reasoning exercise where the required steps (i.e. construction activities) are identified to yield the desired result (i.e. completed building structure). The planning process begins with a result (i.e. a building design). Essential aspects of construction planning include the generation of required activities, and the analysis of the implications of these activities. The programming language Prolog, which stands for Programmation et Logique or Programming in Logic (Sterling and Shapiro, 1988)), has its own reasoning engine. Prolog reasoning engine is a backward-chaining procedure using depth-first search algorithm on ordered facts and rules until a solution is found. The similarity in the reasoning style of Prolog and the required mode of reasoning in construction planning is a strong motivation in investigating the use of Prolog in implementing a software-based workbench for construction planning.

1.2 USE OF IFC

The International Alliance for Interoperability (IAI, visit http://www.iai-international.org/iai_international/) IFC data format provides a catalogue of the building elements within the project and can be useful as the basis for developing a first draft construction schedule. The construction of these elements can be automatically converted into job activities, which can then be automatically sequenced. This will provide a draft construction schedule that can then be used in existing Construction Project Management software, such as Primavera or Microsoft Project for further analysis (see Figure 1). The sequencing information can also be able to be used in 4D CAD (3D visualisation + time) software enabling users to visualise the sequence of construction on the virtual building.

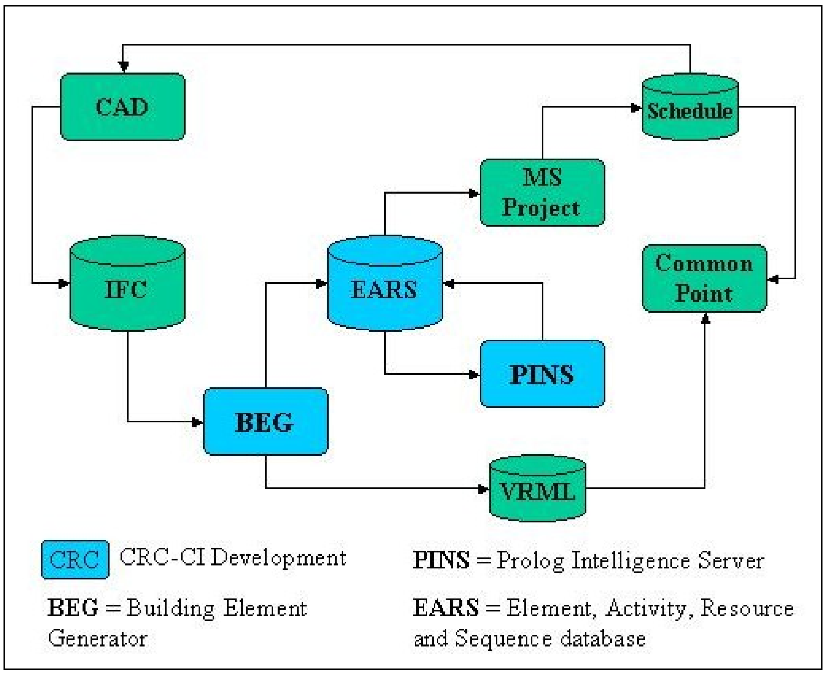

CAD
Schedule
MS Project
IFC
EARS
Common Point
PINS
BEG
VRML
CRC CRC-CI Development
PINS = Prolog Intelligence Server
BEG = Building Element Generator
EARS = Element, Activity, Resource and Sequence database

Figure 1: From 3D CAD to a construction schedule

1.3 SOME DEFINITIONS

- An element is an IFC building elements such as columns, beams and slabs.
- A component is a collection of one or more elements.
- For planning purposes, a construction project is sub-divided into segments or work units called an activity.
- An element or component is associated with one or more activities, resulting to <EA> or <CA> pairs.
- <EA> or <CA> pairs are associated with one or more resources, resulting to <EAR> or <CAR> triples.

2. JOB ACTIVITIES

The segments into which a construction project is subdivided for planning purposes are called activities. The extent a project is subdivided depends on a number of practical considerations. One suggested guideline (Clough and Sear, 1991) is to identify activities by

- Distinct structural elements such as beams, columns, footings or slabs
- Required craft, crew or equipment
- Material such as concrete, timber or steel
- Different sub-contractors

An activity can be a relatively large segment or may be limited to several steps. For example, the construction of a reinforced concrete beam may be considered as a single activity or may consist of construct formwork, place reinforcing steel, pour concrete, cure concrete, strip formwork. The appropriate level of detail depends on several factors such as the nature of project and users (e.g. general contractor or sub-contractor).

A simple way of identifying activities is to assume a one-to-one correspondence between activities and building elements. For example, if there is a beam labelled “beam 101” then there is an activity “construct beam 101” (see Figure 2).

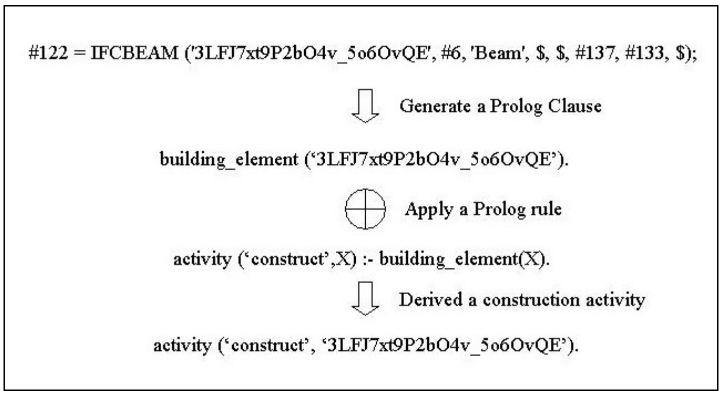
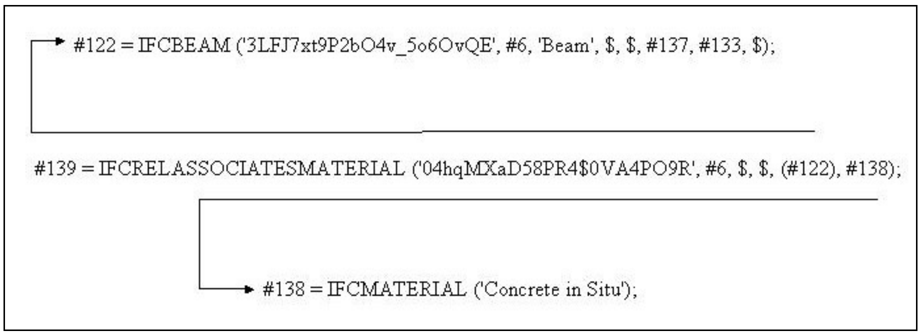

```
#122 = IFCBEAM ('3LFJ7xt9P2bO4v_5o6OvQE', #6, 'Beam', $, $, #137, #133, $);
building_element ('3LFJ7xt9P2bO4v_5o6OvQE').
activity ('construct',X) :- building_element(X).
activity ('construct', '3LFJ7xt9P2bO4v_5o6OvQE').
```

Figure 2: From an IFC building element to a job activity

More detailed job activities can be generated if there is more information about the building elements. For instance, if it is known that a beam is cast in-situ reinforced concrete then the generated activities can be construct formwork, place reinforcing steel, pour concrete, cure concrete, strip formwork.

The method of construction of a particular building element can either be explicitly defined or inferred from the IFC data as illustrated in Figure 3. Once the constructed method is known


```
#122 = IFCBEAM ('3LFJ7xt9P2bO4v_5o6OvQE', #6, 'Beam', $, $, #137, #133, $);
#139 = IFCRELASSOCIATESMATERIAL ('04hqMXaD58PR4$0VA4PO9R', #6, $, $, (#122), #138);
#138 = IFCMATERIAL ('Concrete in Situ');
```
Figure 3: Inferring the construction method from IFC data

3. JOB LOGIC

A construction schedule typically represents a sequence of multiple teams (i.e. trades) that perform individual and distinct work while sharing common workspaces and resources. The logic or rationale behind the sequence of activities in a schedule is referred to as job logic. Job logic includes physical relationship between building elements or components (e.g. a column supports a beam), work team interactions (e.g. concrete work team and carpenters), or safety and code regulations (e.g. workers in a lower level should be protected from activities above them).

The start of some activities obviously depends on the completion of other activities. However, some activities may be independent from other set of activities and may proceed concurrently. Much of job logic follows from well-established work sequences that are standard in the trade. Nevertheless, there is generally more than one approach and no unique order of activities in any significant construction project.

The job logic between construction activities can be divided into fixed logic and soft logic. Fixed logic, such as the relation between installation of the reinforcing bars and pouring of concrete, will not change in any sensible construction process. Soft logic, such as at which end of a bridge should construction start may depend on various factors.

The basic goal of using logic programming is to capture both basic construction principles and local industry practices in order to provide a guideline in generating initial construction schedule. The software architecture of this approach is shown in Figure 4.

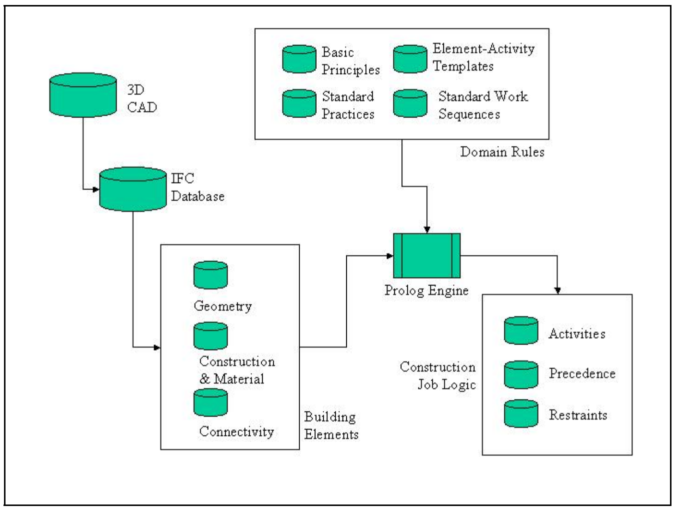


Figure 4: Data-flow diagram for generating construction job logic from IFC data

Consider a trivial example of a structure consisting of 4 pad footings, 8 columns, a stiffened raft (ground slab with 4 edge beams), a suspended beam and slab floor (with 4 beams), 4 beams (with no slabs), a roof and a wall as shown in Figure 5. All elements are of in-situ reinforced concrete construction except for the wall, which is of block work construction.

Observe that the resulting schedule, as shown in Figure 5, allows all in-situ concrete construction to be done continuously. This may be beneficial if a single sub-contractor does all concrete works.

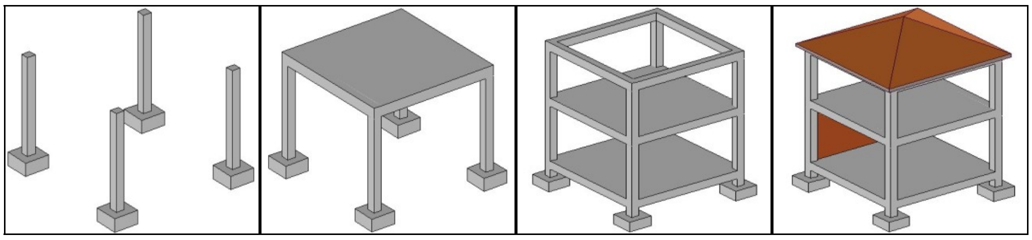


Figure 5: Construction stages of trivial building as an illustrative example

4. PROLOG STRUCTURES

Prolog is a logic language that is particularly suited to programs that involve symbolic or non-numeric computation. It is a frequently used language in Artificial Intelligence where manipulation of symbols and inference about them is a common task (Bratko, 1986). Prolog consists of a series of rules and facts. Additional facts, called derived facts are computed from rules and known (i.e. given) facts.

A Prolog program execution is basically presenting a query and solving it using known rules and facts. For example, consider the following facts about building elements and their connectivity:

```
% Building Element
% element(id,element-type,storey,…)
element(c201,'column',2, …).          % Element c201 is a column in level 2
element(s301,'slab',3,…).             % Element s301 is a slab in level 3
% Element Connection
% connected(id,id)
connected(c201,s301)                  % Element c201 is connected to s301
```

Consider also the following rule about support relation between elements:

```
% Support Relationship
support(X,Y) :-                       % X support Y if …
   element(X,'column',Lc,…),          % X is a column in level Lc
   Ls is Lc+1,                        % Ls is the level just above Lc
   \+ ground_level(Ls)                % Ls is not the ground level
   element(Y,'slab',Ls,…).            % Y is a slab in level Ls
```

The rule above can be interpreted as "If a column is connected to a slab, the column is just below the slab, and the slab is not a ground slab, then the column supports the slab".

Given a query:

```
% Find pairs of (X,Y) such that X supports Y
support(X,Y)?
```

Such a query will result to: X = c201 and Y = s301

Other types of relationships between elements can also be defined using rules in Prolog. For instance, a "constructed together" relation can be written as:

```
% Together Relationship
together(X,Y) :-                      % X is constructed together with Y if
   connected(X,Y),                    % X is connected to Y
   element(X,'beam',…),               % X is a beam
   construction(X,'in-situ RC'),      % X is an in-situ RC beam
   element(Y,'slab',…),               % Y is a slab
   construction(Y,'in-situ RC').      % Y is an in-situ RC slab
```

The rule above can be interpreted as "Beams and slabs in a reinforced concrete floor are constructed together".

Derived relationships such as "support" and 'together" can in turn be use to derived other relationships such as "constructed before".

```
% Precedes Relationship
precedes(X,Y) :-                      % X is constructed before Y if
   support(X,Y),                      % X support Y
   \+ together(X,Y).                  % X is not constructed together with Y.
```

5. ELEMENTS AND ASSOCIATED ACTIVITIES

Associating a set of construction activities to a building element or a group of building elements is a critical step in automatically generating workable job logic. The set of activities associated with a particular building element (i.e. column, slab, beam, or wall) depends on the construction method (i.e. reinforced concrete, steel frame, pre-cast concrete, or composite construction).

For example, the following are activities typically associated with reinforced concrete construction written as Prolog facts:

```
% Construction Activities
activity(1,'formwork').
activity(2,'place reinforcement').
activity(3,'pour concrete').
activity(4,'wait and cure concrete').
activity(5,'strip formwork').
```

Activity templates are used to associate the sequence of activities to a particular combination of building element and construction type. For instance, consider the following Prolog rule and facts:

```
% Element-Activity Pair
element_activity(X,A) :-
   element_activity_list(X,L),
   member(A,L).
% Element-Activity List
element_activity_list(X,L) :-        % L is the list of activities associated with X
   element(X,T,…),                   % T is the element type of X
   construction(X,C),                % C is the construction method of X
   activity_template(T,C,L).

% Activity Templates
activity_template('slab','rc',[1,2,3,4,5]).
activity_template('column','rc',[2,1,3,4,5]).
….
```

Note that for reinforced concrete columns, the activity template specifies that the reinforcement be put in place before the formwork while for slabs the formwork comes before the placement of reinforcement.

5.1 ACTIVITIES AND RESOURCES

Construction resources in traditional construction management views are divided into three categories, namely, labour, material and equipment. These are inadequate for constructability review of the schedule. In recent years, construction space was identified as another important resource type in construction planning. The important role of construction space in construction planning has been illustrated in several studies. Such as:

- Space occupation as a resource constraint (Thabet and Beliveau, 1994)
- Space scheduling model (Riley and Sanvido, 1997)
- Time and space conflicts (Akinci et. al, 2002)

5.2 EQUIPMENT RESOURCES

The definition of construction activities can be extended to include required equipment resources. For example, *activity(3,'pour concrete')* can be extended to *activity(3,'pour concrete','concrete mixer').* Unfortunately, the required equipment may vary depending on the element type and its location. For instance, pouring concrete on a ground slab may also required a trowelling tool/machine while pouring concrete above ground level may require concrete pump and/or crane. Hence, an activity may have two hierarchies of required resources: the minimum resource requirement (i.e. concrete mixer) and the conditional resource requirement (i.e. trowelling tool).

Identifying the required resources for a given element-activity pair is part of the domain knowledge and can be written as Prolog rules as follows:

```
% Element-Activity-Resource Triples
element_activity_resource(X,A,R) :-
   activity(A,_,ResList),
   member(R,ResList).
element_activity_resource(X,A,'trowelling tool') :-
   activity(A,'pour concrete',_),
   element(X,'slab',L),
   ground_level(L).
```

5.3 OTHER RESOURCES

Other traditional resources such as material and labour can be handled in the same way equipment resources are processed as described in the previous section. In fact, even a more abstract concept such as time-space constraints can be dealt with in a similar manner. Consider a fragment of a scheduling exercise shown in Table 1, Table 2 and Table 3.

The schedule fragment in Table 1 shows a construction plan without consideration to resource constraint. It only considers precedence relations (i.e. job logic) between activities. Table 2 presents the revised schedule after incorporating constraint on equipment resource, such as "There is only one crane available." Finally Table 3 illustrates that space constraints can be treated like tangible resources such as equipment and labour.

There are several issues regarding resource constraints that were not considered in this paper, these include: Shared equipment resource, level of resources (i.e. amount of material, number of available equipments, over-time work), and information type resources (i.e. fabrication drawing, detailed engineering design, union contracts)

Table 1: A construction schedule without resource constraint

ID	Task Name	Duration	Start	Finish	Predecessor
9	Erect steel columns	6 days	Tue 8/06/04	Tue 15/06/04	7
10	Erect steel beams	4 days	Wed 16/06/04	Mon 21/06/04	9
11	Erect steel joists	3 days	Tue 22/06/04	Thu 24/06/04	10,8
12	Fireproof structural steel	5 days	Fri 25/06/04	Thu 1/07/04	11
13	Install metal deck	3 days	Fri 2/07/04	Tue 6/07/04	12

ID	Task Name	Duration	Start	Finish	Predecessor
14	Erect precast concrete wall panels	15 days	Fri 25/06/04	Thu 15/07/04	11
15	Pour concrete topping on metal deck	2 days	Wed 7/07/04	Thu 8/07/04	13
16	Build forms for slab-on-grade	3 days	Wed 7/07/04	Fri 9/07/04	13
17	Install reinforcement for slab-on-grade	3 days	Mon 12/07/04	Wed 14/07/04	16
18	Pour concrete in slab-on-grade	2 days	Thu 15/07/04	Fri 16/07/04	17

Table 2: A schedule with constraint on equipment resource

ID	Task Name	Duration	Start	Finish	Predecessors	Resources
9	Erect steel columns	6 days	Tue 8/06/04	Tue 15/06/04	7	crane
10	Erect steel beams	4 days	Wed 16/06/04	Mon 21/06/04	9	crane
11	Erect steel joists	3 days	Tue 22/06/04	Thu 24/06/04	10,8	crane
12	Fireproof structural steel	5 days	Fri 25/06/04	Thu 1/07/04	11	crane
13	Install metal deck	3 days	Fri 2/07/04	Tue 6/07/04	12	crane
14	Erect precast concrete wall panels	15 days	Fri 9/07/04	Thu 29/07/04	11	crane

Table 3: A schedule where space constraint is treated as ordinary resource

ID	Task Name	Duration	Start	Finish	Predecessors	Resources
14	Erect precast concrete wall panels	15 days	Fri 9/07/04	Thu 29/07/04	11	crane, space
15	Pour concrete topping on metal deck	2 days	Wed 7/07/04	Thu 8/07/04	13	crane, space
16	Build forms for slab-on-grade	3 days	Wed 7/07/04	Fri 9/07/04	13	
17	Install reinforcement for slab-on-grade	3 days	Mon 12/07/04	Wed 14/07/04	16	
18	Pour concrete in slab-on-grade	2 days	Fri 30/07/04	Mon 2/08/04	17	space

6. CONSTRAINT LOGIC PROGRAMMING

A constraint can intuitively be thought of as a restriction on a space of possibilities. Constraints arise naturally in most areas of human endeavour. For instance, the three angles of a triangle sum to 180 degrees or the trusses supporting a bridge can only carry a certain static and dynamic load.

Constraint Logic Programming (CLP) is a powerful programming framework with significant applications. The design of the CLP framework is based on the insight that Logic Programming is a constraint-solving algorithm and as such can be combined with various other constraint-solving algorithms (Vlahavas et. al, 1998).

Today CLP is contributing exciting new research directions in artificial intelligence (i.e. natural language understanding, scheduling, planning, configuration), operations research and combinatorial optimization. The field is being driven by the demands of increasingly sophisticated real-world applications, one of which could be construction scheduling.

6.1 WORKFLOW SCHEDULING

Construction scheduling can be expressed as a workflow-scheduling problem, which states, "Find an execution sequence of activities in a workflow such that all constraints are satisfied." A workflow is defined as "a coordinated set of activities that act together to achieve a well-defined goal".

Several research works have addressed the classification of sequencing rationale or constraints. In particular, a CIFE working paper (Koo and Fischer, 2003) investigated existing approaches for classifying and representing project-independent sequencing rationale. The paper focused on physical relationships between building elements or components, workspace interaction, path interference and code regulations. A modified version of their classification is presented in Table 4.

Table 4: Typical constraints in a construction scheduling problem

Type	Constraints	Flexible?	Effect
Physical	Supported by	No (Normally)[a]	Enabling
	Connected to	Yes	Impending
	Covered by	No	Enabling
	Damaged by	No (Normally)[b]	Impending
Resource	Equipment	Yes	Impending
	Labour	Yes	Impending
	Trades	Yes	Impending
Space	Shared Workspace	Yes	Impending
	Obstruction	No	Impending
	Provide access	Yes	Enabling
Code Regulation	Safety	No	Impending
	Inspection	No	Impending
	Testing	No	Impending

[a] - A supported by constraint can be relaxed if temporary support can be provided
[b] - A damaged by constraint can be relaxed if the element or component can be protected

6.2 USING ECLISPE

ECLiPSe provides an excellent tool to create hybrid algorithms to solve large-scale combinatorial optimization problems (LSCO) such as resource constrained scheduling. Combinatorial optimization problems that have the scale and complexity of real-world requirements are interesting because of their commercial importance. LSCO problems are approached from a constraint-programming (CP) angle. Especially, the ECLiPSe CP platform strongly supports such algorithm specialization. One such real world based LSCO is the kernel resource feasibility problem (KRFP), which generalizes most scheduling benchmarks including resource constrained project scheduling. In the KRFP, there are several types of resources and a given quantity of each. There are also a fixed number of activities, which may require a quantity of a specific resource type during its execution. The aim is to schedule activities so that all the constraints are satisfied, and the demand on any resource does not exceed its resource quantity at any time on the scheduling horizon.

Consider a trivial set of construction activities shown in Table 5. Note that all activities except the 4th task require a crane. The best construction schedule will take 23 working days, with the 4th and 6th activities being done in parallel. This scheduling problem can be expressed in ECLiPSe as:

```
solveSchedule(End_date, T1, T2, T3, T4, T5, T6) :-
   Tasks = [T1, T2, T3, T4, T5, T6, Source, Target],
   R1 = resource with [equipment : crane],
   Source = task with [start : 0, duration : 0, need : []],
   T1 = task with [duration : 6, need : [Source], use : [R1]],
   T2 = task with [duration : 4, need : [T1],use : [R1]],
   T3 = task with [duration : 3, need : [T2], use : [R1]],
   T4 = task with [duration : 5, need : [T3]],
   T5 = task with [duration : 3, need : [T4], use : [R1]],
   T6 = task with [duration : 6, need : [T3], use : [R1]],
   Target = task with [duration : 0, need : [T5, T6],start:End_date],
                                                                  ….
```

Table 5: A trivial set of construction activities

ID	Task Name	Duration	Predecessors	Resources
1	Erect steel columns	6 days		crane
2	Erect steel beams	4 days	1	crane
3	Erect steel joists	3 days	2	crane
4	Fireproof structural steel	5 days	3	
5	Install metal deck	3 days	4	crane
6	Erect precast concrete wall panels	15 days	3	crane

7. CONCLUSION

This paper has presented an overview on how logic programming can be use in construction planning, specifically in the generation of a draft construction schedule from 3d CAD model. It described the transformation of a 3d IFC data model into a draft construction schedule and finally into a 4d visualisation.

The authors of this paper would also like to highlight that the work on automatic generation of draft construction schedule from IFC data is far from complete. Much work is yet to be done is developing a more robust domain rule base in order to account for actual construction projects.

The paper also includes the potential use of constraint logic programming in construction schedule generation. This research stream is not part of the CRC supported project.

Acknowledgement: The authors would like to express their appreciation to the CRC for Construction Innovation for the support most of the work described in this paper [Project Number 2002-056-C, Construction Planning Workbench].

8. REFERENCES

Akinci B., Fischer M., Levitt R., Carlson R. (2002) "Formalization and automation of time-space conflict analysis", J Comput Civil Engng, Volume: 16, Issue: 2, pp. 124-134

Bratko, I. (1986) Prolog Programming for Artificial Intelligence, Addison-Wesley Publishers Ltd, ISBN 0-201-14224-4

Calahan, M. Quckenbush, D. and Rowings, J. (1992) Construction Project Scheduling, McGraw Hill Inc. ISBN 0-07-009701-1, p. 289

Chevallier, N.J. and Russell, A.D. (2001) “Developing a draft schedule using templates and rules”, J Construction Engineering and Management, September-October, pp. 391-398

Clough, R.H. and Sear, G.A. (1991) Construction Planning Management, John Wiley & Sons, ISBN 0-471-54608-9, page 55.

Fischer M., Froese T. (1996) "Examples and characteristics of shared project models", J Computing in Civil Engineering, Volume: 10, Issue: 3, pp. 174-182

Koo, B. and Fischer, M. (2003) Formalizing construction sequencing constraints for rapid generation of schedule alternatives, A CIFE Working Paper #75, Stanford University

Riley D.R., Sanvido V.E. (1997) "Space planning method for multistorey building construction", J Construction Engineering Management, Volume: 123, Issue: 2, pp. 171-180

Sterling, L. and Shapiro, E. (1988) The Art of Prolog, MIT Press, ISBN 0-262-19250-0.

Thabet W.Y., Beliveau Y.J. (1994) “Modelling work space to schedule repetitive floors in multistorey buildings", J Construction Engineering Management, Volume: 120, Issue: 1, pp. 96-116

Vlahavas, I., Tsarchapoulos, P. and Sakellariou, I. (1998). Parallel and Constraint Logic Programming, Kluwer Academic Publishers, The Netherlands