Prof. Banaszak Z., M. Sc. Bocewicz G.
Technical University of Koszalin, Poland
e-mail: banaszak@tu.koszalin.pl

TOWARDS UNIFIED FRAMEWORK FOR DEDICATED DSS DESIGN

A unified framework standing behind of a methodology aimed at object oriented decision support system design is considered. First of all the consistency of the assumed knowledge bases describing an object (enterprise) and requests (standard options supporting a decision maker), respectively are examined. Then the knowledge base representation is transformed into representation of so called constraint satisfaction problem (CSP). Possible ways of the CSP decomposition as well as possibility of different programming languages application lead then to a problem aimed at searching for a distribution strategy allowing one to interact in an on-line mode.

MODEL REFERENCYJNY PROCEDURY PROJEKTOWANIA DEDYKOWANYCH SYSTEMÓW WSPOMAGANIA DECYZJI

Streszczenie. *Praca przedstawia zarys metodyki projektowania obiektowo zorientowanych systemów wspomagania decyzji. Wyjściowa baza wiedzy składająca się z baz opisujących odpowiednio obiekt (przedsiębiorstwo) i zbiór kontekstowo osadzonych pytań (standardowych opcji wspomagających zarządzanie) testowana jest pod kątem spójności i niesprzeczności. Pozytywny wynik testu (gwarantujący adekwatność opcji i obiektu) pozwala przejść do reprezentacji problemu spełniana ograniczeń. Różne dekompozycje tego modelu, uwzględniające różne platformy programowania pozwalają sformułować problem poszukiwania efektywnej (gwarantującej interaktywną pracę systemu wspomagania) strategii dystrybucji zmiennych decyzyjnych.*

1. INTRODUCTION

Managers need to be able to utilize a modern decision support tools as to undertake optimal business decisions in further strategic perspective of enterprise operation. However, commercially available software packages employing the methods based on local search metaheuristics such as simulated annealing, tabu search, genetic algorithms, are quite costly and require skilful and well trained personnel. Moreover, they are not able to integrate (to treat in an unified way) such different tasks as production and transportation routings, production and batch sizing as well as tasks scheduling [5].

In that context our objective is to provide a constraint programming based methodology aimed at designing of task oriented decision support systems (DSS). In other words, the framework we are looking for should be able to cope with a problem defined in terms of finding of a feasible schedule that satisfies the constraints imposed by the duration of production order processing, the cost assumed, and the time-constrained resources availability.

Often repeating requests regard the questions such as: Whether in a given enterprise employed with the machine tools, automated guided vehicles (*AGVs*), buffers and warehouses a production order submitted can be completed due assumed period of time? Can the consumer's requirements regarding the final cost production be guaranteed? Does a given number of transportation means guarantee due time product delivery? Is the production capacity of the company sufficient to accept a new production order? Is the company able to respond? How to obtain such a response in an on-line mode? What strategy of production order processing is the most efficient one? Can the consumer's requirements be fulfilled within the assumed Extended Enterprise structure? Does the assumed set of SMEs guarantee a resultant Extended Enterprise to accomplish a given production order?

Respond to the questions usually involve many different aspects and contexts, e.g., money flow, personnel and/or resources allocation, tasks scheduling, workflows planning, and so on. In that context, the Constraint Programming/Constraint Logic Programming (*CP/CLP*) languages by employing the constraints propagation concept and by providing unified constraints specification can be considered as a well-suited framework for development of decision making software aimed the small and medium sized enterprises (*SMEs*) [1]. Because of their declarative nature, for a use that is enough to state *what* has to be solved instead *how* to solve it [4] the approach seems to be very friendly for modelling of a company real-life and day-to-day decision-making [6].Respond to the questions usually involve many different aspects and contexts, e.g., money flow, personnel and/or resources allocation, tasks scheduling, workflows planning, and so on.

2. PROBLEM STATEMENT

Given knowledge base representation of a *SME*, and knowledge base of context-oriented queries. The *SME's* specification includes parameters describing the parameters such as the number of resources available, their efficiency, capacity, etc., as well as relations linking particular workstations, pallets, tools and so on. In turn, the queries encompassing the standard options of *SME* management are specified by data relevant to a production order requirements and the enterprise capability. The objective is to fined a *DSS* allowing one to respond to the any question related to the *SME* considered in an interactive mode. So, the problem we are facing with regards of a question: Whether for the commercially available programming languages there exists a way enabling one to evaluate a possibility of relevant *DSS* design? The graphical illustration of the problem considered is shown in Fig. 1.

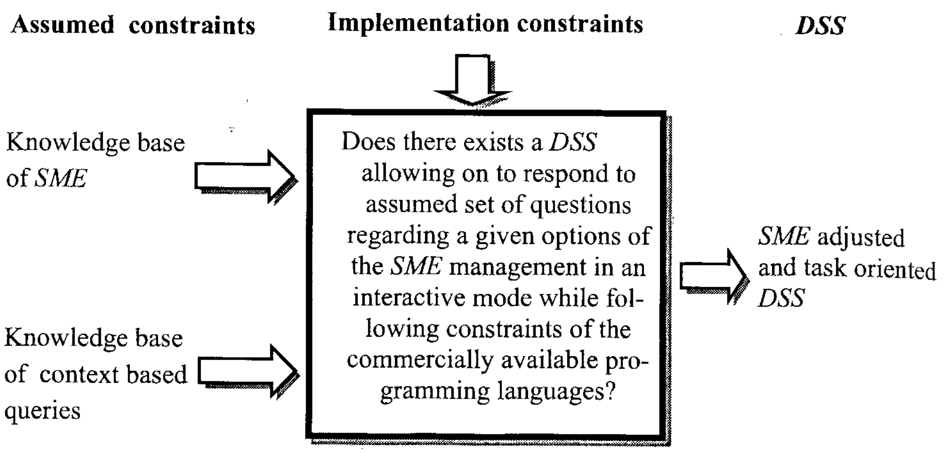

Assumed constraints
Implementation constraints
DSS
Knowledge base of SME
Knowledge base of context based queries
Does there exists a DSS allowing on to respond to assumed set of questions regarding a given options of the SME management in an interactive mode while following constraints of the commercially available programming languages?
SME adjusted and task oriented DSS

Fig.1 Illustration of the problem statement

3. KNOWLEDGE BASE REPRESENTATION

It is assumed that any system can be specified in terms of knowledge base composed of facts and rules determining system's properties and linking them relations, respectively. Formally, knowledge base RW is defined as a pair: $RW = < \alpha, F(\alpha)>$, where

$\alpha = (\alpha_1, \alpha_2, \dots, \alpha_N)$ – is a sequence of elementary formulas specifying system's properties; α_i – is the i-th assertion (specified in terms of binary logic), and $a_i = w(\alpha_i) \in \{0,1\}$ is a logic value of the assertion α_i

$F(\alpha) = \{F_1(\alpha), F_2(\alpha), \dots, F_K(\alpha)\}$ – is a sequence of facts specifying relations among properties (in terms of logic operators: conjunction, disjunction, negation, and implication); $F_j(a)$ – states for a binary value of expression $F_i(\alpha)$.

So, to any $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_N)$ corresponds $a = (a_1, a_2, \dots, a_m)$. Consequently $(a_1, a_2, \dots, a_m)$ states for a sequence of values associated to $w(\alpha)$.

In any system description the following categories can be distinguished:

$\alpha_x = \{\alpha_{x1}, \alpha_{x2}, \dots, \alpha_{xk}\}$ – a set of elementary formulas specifying so called input system variables, $\alpha_{xu} \in \{\alpha_1, \alpha_2, \dots, \alpha_N\}$

$\alpha_y = \{\alpha_{y1}, \alpha_{y2}, \dots, \alpha_{yp}\}$ – a set of elementary formulas specifying so called output system variables, $\alpha_{yu} \in \{\alpha_1, \alpha_2, \dots, \alpha_N\}$

$\alpha_w = \{\alpha_{x1}, \alpha_{x2}, \dots, \alpha_{xr}\}$ – a set of auxiliary elementary formulas specifying system functioning, $\alpha_{wu} \in \{\alpha_1, \alpha_2, \dots, \alpha_N\}$

Of course, $\alpha_x \cup \alpha_y \cup \alpha_w = \alpha$, $\alpha_x \cap \alpha_y = \emptyset$, $\alpha_x \cap \alpha_w = \emptyset$, $\alpha_y \cap \alpha_w = \emptyset$, and $a_x = \{a_{x1}, a_{x2}, \dots, a_{xk}\}$, $a_y = \{a_{y1}, a_{y2}, \dots, a_{yp}\}$, $a_w = \{a_{x1}, a_{x2}, \dots, a_{xr}\}$, so $a = (a_{x1}, a_{x2}, \dots, a_{xk}) \wedge (a_{y1}, a_{y2}, \dots, a_{yp}) \wedge (a_{x1}, a_{x2}, \dots, a_{xr})$ corresponds to $\alpha = (\alpha_{x1}, \alpha_{x2}, \dots, \alpha_{xk})\wedge(\alpha_{y1}, \alpha_{y2}, \dots \alpha_{yp})\wedge(\alpha_{x1}, \alpha_{x2}, \dots, \alpha_{xr})$.

$F_x(\alpha_x) = \{F_{x1}(\alpha_x), F_{x2}(\alpha_x), \dots, F_{xP}(\alpha_x)\}$ – set of input facts, i.e. assertions describing properties of system input,

$F_y(\alpha_y) = \{F_{y1}(\alpha_y), F_{y2}(\alpha_y), \dots, F_{yR}(\alpha_y)\}$ – set of output facts, i.e. assertions describing properties of system output.

Of course, besides of the real objects such as SMEs the above representation can be applied to any other objects, e.g., constraints, specifications, etc.

4.CONSTRAINT SATISFACTION PROBLEM REPRESENTATION

The declarative character of Constraint Programming languages and a high efficiency in solving combinatorial problems creates an attractive alternative for the currently available (employing operation research techniques) systems of computer-integrated management [2].

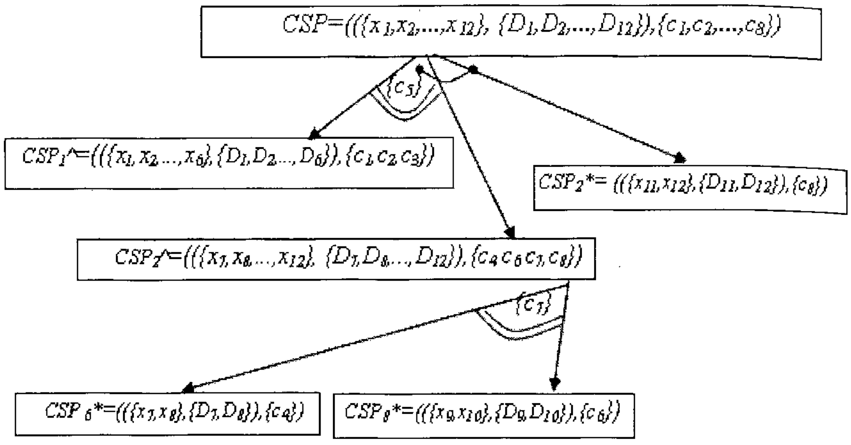
The Constraint Satisfaction Problem $CSP = ((X,D,),C)$ consists of a set of variables $X = \{x_1, x_2,\dots,x_n\}$, their domains $D = \{D_i \,|D_i = [d_{i1},d_{i2},\dots,d_{ij},\dots,d_{im}],\ i = \{1,\dots,n\}\}$, and a set of constraints $C = \{C_i \,|i = \{1,\dots,L\}\}$. A solution is such an assignment of the variable values that all the constraints are satisfied.

In general case any *CSP* may be decomposed (see Fig.2), however, either into a set of loosely coupled problems or into a set of strongly coupled problems. Possible ways of CSP decomposition enable one to take into account the real life constraints following from:

- a way of a problem specification (i.e., a set of elementary problems recognized)
- a programming language implemented (some structures of dependent problems may or may not be accepted by *CP*/*CLP* packages)
- a way of a *CSP* resolution (e.g., the loosely coupled subproblems can be computed independently within an multiprocessor environment)
- a searching strategy applied (the order of subproblems resolution results in a CSP makespan).

The above observation leads to a concept of a reference model of a *CSP* decomposition, [4]. So, since each subproblem corresponds to a standard constraint problem structure: *(({a set of decision variables}. {a set of variable domains}), {a set of constraints})*, hence some AND/OR – like graph representation can be used both in the course of analysis of the *CSP* programming and its resolution.

It should be noted that any knowledge base can be represented in terms of $CSP = ((a,D), \{F(a)=1\})$, where $D = \{D_i \mid D_i = \{0,1\}, i = 1..N\}$, $F(a)=1$ a sequence of facts: $(F_1(a)=1, F_2(a)=1, \dots, F_K(a)=1)$.


CSP=((({x_1,x_2,...,x_12}, {D_1,D_2,...,D_12}),{c_1,c_2,...,c_8})
{c_5}
CSP_1^=((({x_1, x_2 ...,x_6},{D_1,D_2...,D_6}),{c_1 c_2 c_3})
CSP_2*= ((({x_11,x_12},{D_11,D_12}),{c_8})
CSP_2^=((({x_7, x_8...,x_12}, {D_7,D_8...,D_12}),{c_4 c_6 c_7, c_8})
{c_7}
CSP_6*=((({x_7,x_8},{D_7,D_8}),{c_4})
CSP_8*=((({x_9,x_10},{D_9,D_10}),{c_6})

Legend:

CSP_2*, CSP_6*, CSP_8* - elementary subproblems,
CSP_1^, CSP_2^, CSP_2*, CSP_6*, CSP_8* - strongly coupled subproblems,
CSP_1*, CSP_2* $= ((\{x_1\text{-}x_{10}\},\{D_1\text{-}D_{10}\}),\{c_1\text{-}c_7\})$ - loosely coupled subproblems,
$\curlyvee$ - decomposition into dependent subproblems,
•⌣• - decomposition into loosely coupled subproblems.

Fig.2 Illustration of the *CSP* problem decomposition.

5. FEASIBLE SOULTIONS

The question considered regards of $F_x(\alpha_x)$ following the implication: $F_x(\alpha_x) \Rightarrow F_y(\alpha_y)$. In other words, the question is: What are α_x and $F_x(\alpha_x)$, if either, ensuring the system property $F_y(\alpha_y)$.

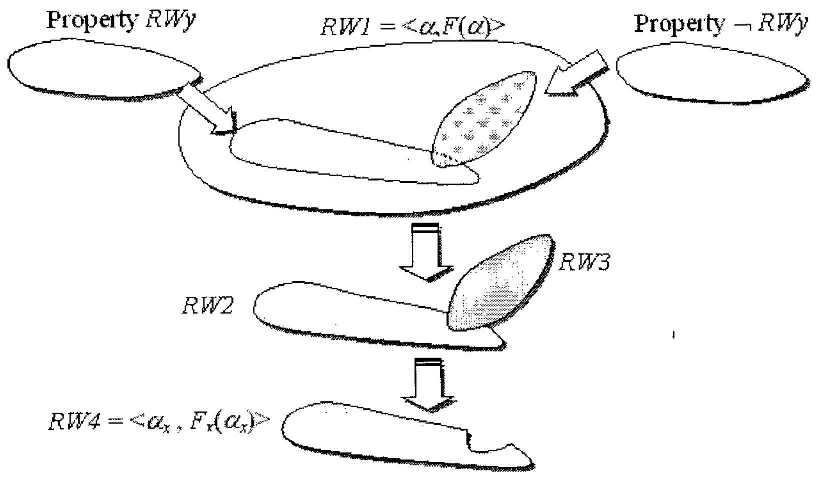
Consider the knowledge base $RW'=<\alpha',F'(\alpha)>$ corresponding to a system considered and the knowledge base stating a question $RW'' = <\alpha'',F''(\alpha)>$, e.g. regarding a given system's property. The resultant knowledge base $RW1 = <\alpha,F(\alpha)>$ (see Fig. 4) provides a framework for the considered problem statement: Does there exist $F_x(\alpha_x)$, ensuring $F_y(\alpha_y)$ holds? More precisely, in order to determine the feature $F_x(\alpha_x)$ a set of facts following feature $F_y(\alpha_y)$ while do not following $\neg F_y(\alpha_y)$ have to be determined.

The scheme of the searching procedure is shown in Fig.3. It means the knowledge base $RW1$ including the conditions implying $F_y(\alpha_y)$ as well as the knowledge base $RW2$ non including the conditions implying $F_y(\alpha_y)$ are refined from the knowledge base considered RW.

The knowledge bases obtained enable to determine the final knowledge base $RW3$, i.e. modified $RW1$ (not including elementary formulas and facts included in $RW2$.

In order to implement the above procedure in terms of logic-algebraic method the sets of binary values S_a, S_x, and S_y following a, a_x, and a_y, (while corresponding to $F(\alpha)$, $F_x(\alpha_x)$, and $F_y(\alpha_y)$) have to be defined due to formulas: $S_a = \{a\colon F(a) = 1\}$, $S_x = \{a_x\colon F_x(a_x) = 1\}$, $S_y = \{a_y\colon F_y(a_y) = 1\}$

Assumption all the facts of *RW* are true implies that among sequences α there are also such for which $F(a) = 1$ holds, see $S_a = \{a\colon F(a)=1\}$. The associated set S_a guaranteeing the facts describing the system are true can be treaded as *RW*. Searching for the set S_x representing $F_x(\alpha_x)$ which can be treated as *RW3* requires two subsets, i.e. S_{x1} corresponding to *RW2* while following $F_y(a_y)=1$, and S_{x2} corresponding to *RW2* while following $F_y(a_y)=0$.



RW1 – knowledge base following $F_y(\alpha_y)$,
RW2 – knowledge base following $\neg F_y(\alpha_y)$,
RW3 – knowledge base containing conditions sufficient for $F_y(\alpha_y)$.

Fig.3 Sufficient conditions refinement.

Finally, $S_x = S_{x1} \setminus S_{x2}$, where S_{x1}, and S_{x2} are determined for a_x from equations :
for S_{x1}:

$$\begin{cases} F(a) = 1 \\ F_y(a_y) = 1 \end{cases} \quad (1)$$

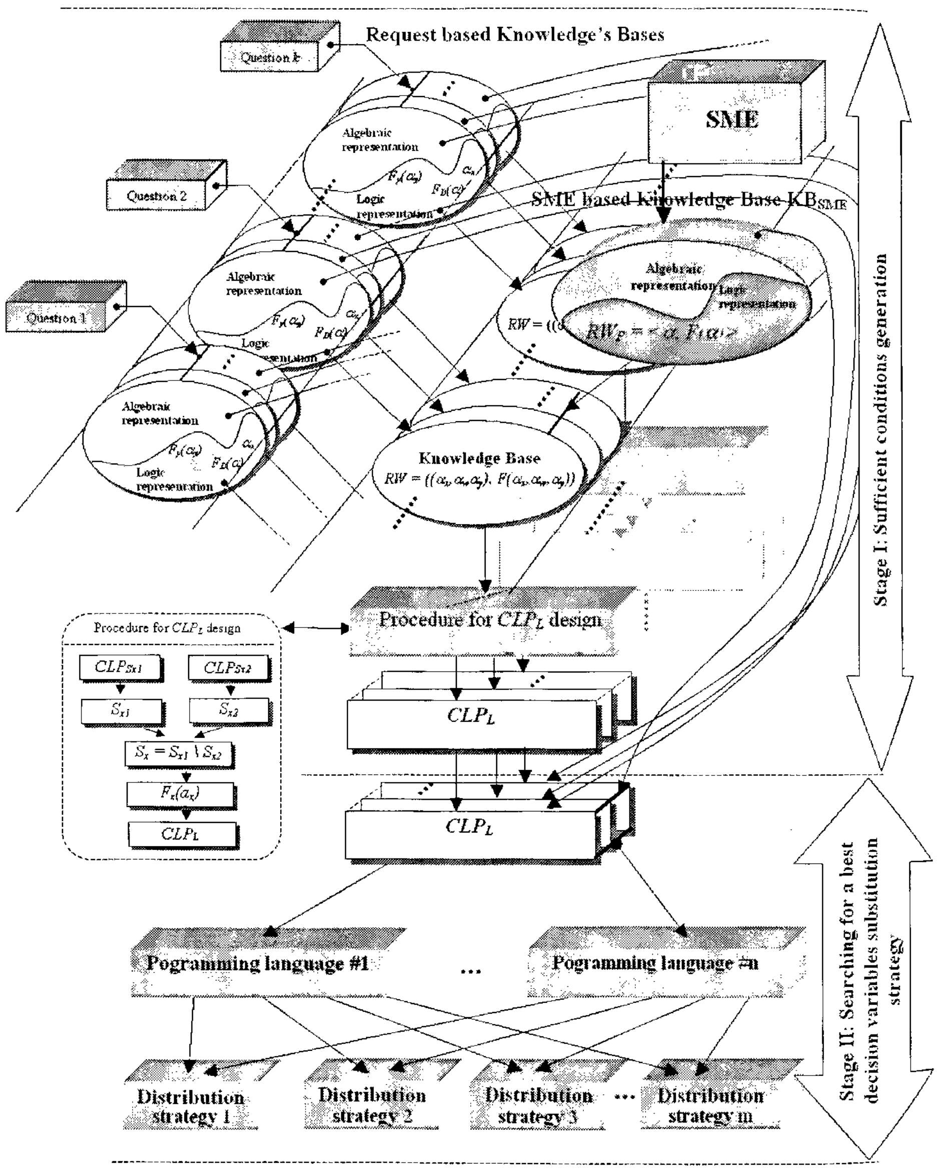
for S_{x2}:

$$\begin{cases} F(a) = 1 \\ F_y(a_y) = 0 \end{cases} \quad (2)$$

where: $F(a)=1$ stands for the set of facts: $\{F_1(a)=1, F_2(a)=1, \dots, F_K(a)=1\}$

6. METHODOLOGY FOR INTERACTIVE DECISION SUPPORT SYSTEMS DESIGN

The proposed methodology consists of two stages, Fig. 4. Due to the first one the CLP_L including the sufficient conditions (i.e. guaranteeing a solution there exists) is provided.


Request based Knowledge's Bases
Question k
Question 2
Question 1
SME
SME based Knowledge Base KB_SME
Algebraic representation
Logic representation
$F_y(\alpha_y)$
$F_D(\alpha)$
α_x
$RW_P = < \alpha, F(\alpha) >$
Knowledge Base
$RW = ((\alpha_x, \alpha_w, \alpha_y), F(\alpha_x, \alpha_w, \alpha_y))$
Procedure for CLP_L design
CLP_{Sx1}
CLP_{Sx2}
S_{x1}
S_{x2}
$S_x = S_{x1} \setminus S_{x2}$
$F_x(a_x)$
CLP_L
Pogramming language #1
Pogramming language #n
Distribution strategy 1
Distribution strategy 2
Distribution strategy 3
Distribution strategy m
Stage I: Sufficient conditions generation
Stage II: Searching for a best decision variables substitution strategy

Legend:

$a = a_x \cap a_y \cap a_w$, $a = (a_1, a_2, \ldots, a_m)$, $a_i = w(\alpha_i)$, $D = \{0,1\}$, $C = \{C_P, C_D, C_{Wr}\}$, $C_P = \{C_1, C_2, \ldots, C_K\}$, $C_D = \{C_{D1}, C_{D2}, \ldots, C_{DH}\}$, $C_i = (F_i(a)=1)$, $C_{Di} = (F_{Di}(a)=1)$, $CLP_{Sx1} = ((a_x, D), \{C_P, C_D, F_y(a_y)=1\})$, $CLP_{Sx2} = ((a_x, D), \{C_P, C_D, F_y(a_y)=0\})$,

$S = \{a_x \in a: F_y(a_y) = 1\}$ – set of sequences a_x, following the constraint $F_y(a_y) = 1$,

$F_x(a_x)$ – sufficient conditions treated as an input fact, $C_{Wr} = (F_x(a_x)=1)$ – the constraint following from the sufficient conditions obtained, $CLP_L = ((a, D), \{C_P, C_D, C_{Wr}\})$ – problem specification including the sufficient conditions

$CLP = ((x, D_x), \{C_P, C_D, C_{Wr}, C_N\})$ – problem specification including the sufficient conditions and extended for algebraic, ones, $x = x_N \cap a$ – decision variables vector, where x_N – belongs to the domain D_X of so called algebraic variables

Fig.4 Methodology for interactive decision support systems design

As the input data the *SME* based knowledge base KB_{SME} and the request based knowledge base KB_R are considered. Of course, the different request based knowledge bases may result in different sets of sufficient conditions. This observation provides a way of the sufficient conditions refinement, i.e. a way of DSS adjustment.

The CLP_L extended for other kinds of variables and constraints (so called algebraic ones) results in *CLP* problem. So, due to the second stage a programming languages as well as decision variables substitution strategy guaranteeing interactive usage of the *DSS* designed is provided [4].

Both stages are based on the *CP/CLP* languages. The key point of the methodology proposed regards of the procedure for CSP_L design. In order to illustrate this procedure partially introduced in the Section 5, let us consider the following example.

Consider controller composed of two switches P_1 i P_2. The room's temperature is controlled by the set up relevant switches

- If P_1 is turn on and P_2 is turn off, then the room is wormed up to 20 ^{0}C.
- If P_1 is turn off and P_2 is turn on, then the room is wormed up to 30 ^{0}C.
- If both P_1 and P_2 are turn on, then the room is wormed up to 40 ^{0}C.

The question is: What switches set up guarantee the room's temperature is between 20 and 40 ^{0}C.?

The controller's knowledge base representation is: ***RW* = <α, *F*(α)>:**

$\alpha = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5)$, where:

α_1: „switch P_1 is turn on",

α_2: „switch P_2 is turn off",

α_3: „the room is wormed up to 20 ^{0}C",

α_4: „the room is wormed up to 30 ^{0}C",

α_5: „the room is wormed up to 40 ^{0}C"

α_6: „ the room is not wormed"

$F(\alpha) = (F_1(\alpha), F_2(\alpha), F_3(\alpha), F_4(\alpha))$, where:

$F_1(\alpha)$: $\alpha_1 \wedge (\neg\alpha_2) \Leftrightarrow \alpha_3$

$F_2(\alpha)$: $(\neg\alpha_1) \wedge \alpha_2 \Leftrightarrow \alpha_4$

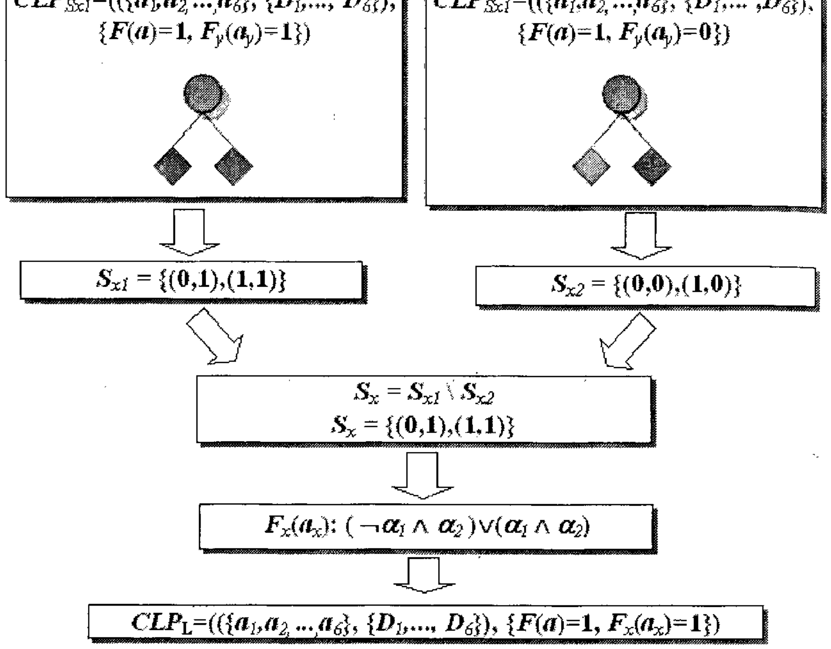
$F_3(\alpha)$: $\alpha_1 \wedge \alpha_2 \Leftrightarrow \alpha_5$

$F_4(\alpha)$: $(\neg\alpha_1) \wedge (\neg\alpha_2) \Leftrightarrow \alpha_6$

Elementary input and output formulas are: $\alpha_x = (\alpha_1, \alpha_2,)$, $\alpha_y = (\alpha_3, \alpha_4, \alpha_5, \alpha_6)$

Required output property is: $F_Y(\alpha_y) = \alpha_4 \vee \alpha_5$

The Fig.5 illustrates the logic-algebraic method based procedure for to the CSP_L design. Note that the procedure follows the scheme of the sufficient conditions refinement shown in Fig. 3.


CLP_{Sx1}=(({a_1,a_2,...,a_6}, {D_1,..., D_6}), {F(a)=1, F_y(a_y)=1})
CLP_{Sx1}=(({a_1,a_2,...,a_6}, {D_1,... ,D_6}), {F(a)=1, F_y(a_y)=0})
S_{x1} = {(0,1),(1,1)}
S_{x2} = {(0,0),(1,0)}
S_x = S_{x1} \ S_{x2}
S_x = {(0,1),(1,1)}
F_x(a_x): (¬α_1 ∧ α_2)∨(α_1 ∧ α_2)
CLP_L=(({a_1,a_2,...,a_6}, {D_1,..., D_6}), {F(a)=1, F_x(a_x)=1})

Legend:

$a_i = w(\alpha_i)$ – decision variable determining logic value of the formulae α_i,

$D_i = \{0, 1\}$ – binary domain of the variable a_i,

$F(a)=1$ – the constraint guaranteeing the all facts F_i hold: $(F_1(\alpha)=1, F_2(\alpha)=1, F_3(\alpha)=1, F_4(\alpha)=1)$,

$F_y(a_y)=1$ – the constraint guaranteeing, the output fact is true,

$F_y(a_y)=0$ – the constraint guaranteeing, the output fact is false,

$F_x(a_x)$ – the input fact, i.e. resultant sufficient condition.

Fig. 5 Illustration of the logic-algebraic method based procedure for to the CSP_L design.

7. CONCLUSIONS

A *CP/CLP* – based modeling framework driven by the logic-algebraic method provide a good platform for development of the task oriented *DSS*. The discussion provided has shown the versatility of *CP/CLP* paradigm for the decision making problems. Possible applications of logic-algebraic method to the examination of sufficient conditions ensuring assumed system's properties as well as the consistency checking techniques greatly reducing the search space and supported by *CP/CLP* prove their efficiency for resolution of the project-driven manufacturing tasks.

Therefore, the proposed approach can be considered as a contribution to project-driven production flow management applied in make-to-order companies as well as for prototyping of the virtual enterprises. That is especially important in the context of a cheap and user-friendly decision support for the *SMEs*. Further research is aimed on the development of the task oriented searching strategies, implementation of which could interface a decision maker with a user-friendly intelligent support system.

References

[1] Banaszak, Z., Zaremba, M., Muszyński, W., 2005, CP-based decision making for SME. *Preprints of the 16th IFAC World Congres*, 3 – 8 July, 2005, Prague, Czech Republic, Eds P. Horacek, M. Simandl, P.Zitek, DVD

[2] Banaszak, Z., Zaremba, M., 2004, CLP-based project-driven manufacturing. *Prep. of the 7th IFAC Symposium on Cost Oriented Automation*, June 6-9, 2004, Gatineau, Quebec, Canada, pp.269-274.

[4] Banaszak, Z., Józefczyk, J., 2005, *Towards CLP-based task oriented DSS for SME,* Applied Computer Science and Production Management, Vol.1, No.1: 161-180.

[5] Barták R., 2003, *Constraint-based scheduling: An introduction for newcomers,* Preprints of the 7th IFAC Workshop on Intelligent Manufacturing Systems, 6-8 April, 2003, Budapest, Hungary, pp.75-80.

[6] Mika, Marek; Waligóra G. Węglarz J., 2003, Metaheuristic approach to the multi-mode resource-constrained project scheduling problem with discounted cash flows and progress payment. In: *Project-Driven Manufacturing*. Banaszak Z., Józefowska J. Eds, WNT, Warsaw, pp.34-47.

[7] Rossi F., 2000, Constraint (Logic) programming: A Survey on Research and Applications, K.R. Apt et al. (Eds.), *New Trends in Constraints*, LNAI 1865, Springer-Verlag, Berlin, pp. 40-74.

[8] Tomczuk I., Banaszak Z., 2004, Constraint programming approach for production flow planning, *Proc. of the 6th Workshop on Constraint Programming for Decision and Control*, pp. 47-54.

[9] Van Hentenryck P., Perron L., Puget J., 2000, *Search and Strategies in OPL*, ACM Transactions on Computational Logic, Vol. 1, No. 2, pp. 1-36.

[10] Wallace M., 2000, *Constraint Logic Programming,* Ed. Kakas A.C., Sadri F., Computat. Logic, LNAI 2407, Springer-Verlag, Berlin, Heidelberg, pp. 512-532.