電子回路の設計知識の記述について

長澤 勲　九大中央計数施設

1. はじめに

本論文では、電子回路の一領域を対象に設計者の知識の定式化を試みた。

電子回路や機械などの設計者の知識は、物理的拘束、設計上の規約、経験等の複雑な集合体であることが想像される。しかし計算機の上にこれを表現しようとするとき形式化の容易さから見て、いくつかの段階に分けることも必要であろう。筆者は次の3つのレベルに分けるのが適当であると考えている。

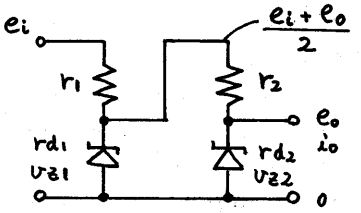
(1) 拘束記述……デバイスの仕様，構造，性能などがどのような拘束関係にあるか。

(2) 解析的記述……デバイスがどのような原理でどのように動作するかの様々な記述。

(3) 構成的記述……デバイスはどのような考えで構成されたか。

以下電子回路を例に説明しよう。(1)は主として設計公式に代表される知識であり，経験規則や(2)の動作の理解から導かれたと見ることができる。設計過程においては、このレベルの知識を用いるのが最も容易である。手続やプロダクション・ルールを用いる代りに拘束を直接解く方法が最近の論理プログラミング［Kowalski 74］によって提案されており、これを用いると多目的性、たとえば、設計、設計の検証、さらに仕様の部分的変更が設計されたデバイスのどの部分に影響をよえるかが調べられるなど、が得られる。(2)は、デバイスの動作の理解が対応する。これには多くのレベルがあり、電子回路では、(a)回路の動作を正確な素子モデルを用

ツェナダイオードによる定電圧回路



安定化定数 $SV \fallingdotseq \dfrac{rd_1}{r_1}\dfrac{rd_2}{r_2}$

出力電圧　$e_o \fallingdotseq v_{z2}$

出力抵抗　$RO \fallingdotseq rd_2$

抵抗値　$r_2 = (e_i - e_o)/2(i_o + i_{z2})$

抵抗値　$r_1 = (e_i - e_o)/2(i_o + i_{z1} + i_{z2})$

但し

rd_1　ツェナダイオード動作抵抗

rd_2　〃

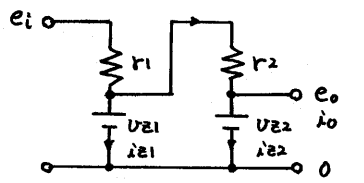
v_{z1}　ツェナ電圧

v_{z2}　〃

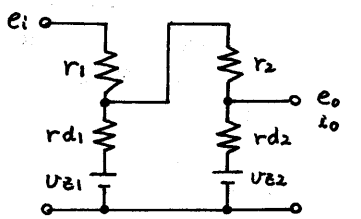
i_{z1}　ツェナ電流

i_{z2}　〃

(図 1.1) 拘束記述



r_1 r_2 計算のための等価回路



回路定数算出のための等価回路

(図 1.2) 解析的記述

[目的] 5V を改善する。
[方法] ツェナダイオード定電圧回路 を従続接続する。

(図1.3)構成的記述

いて数値的に評価し、回路動作の確認や素子定数決定を行うシミュレーション (b)近似的等価回路やデバイスモデルから解析的に動作を説明し、回路動作の主要な成分が何であるかを明らかにすること。(これには回路動作の定性的理解が前提になっている。) (c)回路動作の定性的説明を定性的因果律や拘束によってよえること。しかし定性的因果律は近似の前提となっている暗黙の仮定が多く含まれていて、完全な論理にはなっていないことが知られている。[kleer79] 即ちこの説明は回路動作の要約的記述というべきである。 (3)は、デバイスの構成過程を理解することを意味する。 design heuristics とは目的とする定性的動作を予想してそのような動作を引き起すべく回路の構成や修正を行う規則であるとしよう。しかし、このようにして近似に基づく規則によって構成した回路が本当に目的通りに動作するか否かは、実験や解析を通して確認する必要がある。従ってこのレベルの知識は必ずしも完全ではない。

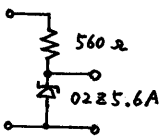
さて以上のように設計知識は多面的であるが、まず (1) のレベルの拘束に注目すべきであろう。以下本論文では主として拘束による記述について検討し、(2) (3) については今後の問題として少しふれるにとどめる。

2. 構造物の表現

図2.1は、ツェナダイオードによる定電圧回路であるが、これほど簡単な回路でも多くの部分や属性を持つことがわかる。知識工学の分野では、構造物を自然に表現する手段としてフレームが用いられてきた。これは知識を適当な大きさにまとめて記述する必要があるからである。フレームにそえられるさまな機能は遺伝階層と手続附加であろう。しかし手続付加には大きな問題があり、論理的基礎付を欠くという批判があり [Hayes 77] はフレームを論理によって支持することを提案している。また [Shank 81] は言語理解と問題解決の2つの研究領域に共通に使える表現としてフレームを位置付け、フレームの持つ、表現の自然さ、知識の適当な大きさへの統合能力と、論理の持つ意味表現の正確さ、module 性を合せ持つ表現の必要性を主張している。この議論は本稿で考えている知識システムにも成立し、ここでは Horn 節による解釈を行う。Horn 節を用いた定理証明システムは手続的解釈が成立し、記述

(仕様)　入力電圧　10V
　　出力電圧　5.6V
　　出力電流　3mA

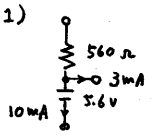
(回路)



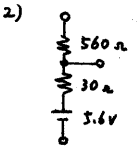
(回路定数)

安定化定数	0.05
出力抵抗	30Ω
温度係数	0.03%

(等価回路1)



(等価回路2)



(図2.1) ツェナダイオードによる定電圧回路

能力が大きいという特徴がある。

まずフレームを論理に解釈する場合各部分またはスロットを各々独立に2項関係と解釈するのが良いか、いくつかの属性をまとめて多項関係と解釈するのが良いかが問題になる。フレームを2項関係集合と解釈する方法は[Deliyani 79]によって提案されている。この方法はISA階層を自然に論理的含意関係におきかえられる長所があるが次のような問題点がある。オームの法則を例にとる。簡単のため抵抗器の属性として端子電圧、端子電流、抵抗値のみを考えよう。電圧、電流、抵抗値をもめるため次の3つの節が必要になる。

(VOLT x v) ← (REG x), (AMP x i), (Reg x r),
(MUL i r v).

(AMP x i) ← (REG x), (VOLT x v), (Reg x r),
(MUL i r v).

(Reg x r) ← (REG x), (VOLT x v), (AMP x i),
(MUL i r v).

これを用いて抵抗値1Ω、電流1Aの抵抗器R1の電圧をもめるには、正節として

(AMP R1 1) ← .
(REG R1) ← .
(Reg R1 1) ← .

を主張し、負節 ← (VOLT R1 v) をかえて導出を行えば良い。(導出法として、LUSH [Hill 74][Kowalski 74]を用いた言語PROLOG[Warren 77]の用語を借りることにしよう。) この方法は節の数が増加すること、もし正節としてたとえば、(Reg R1 1) ← . がよえられてなかったとすると導出が終らない、という欠点がある。すように一般には何を正節に、何を負節にすれば良いか自明でないことも多い。一方4項関係を採用すると単一の節によって多目的な手続が得られる。

(REG x i r v) ← (MUL i r v).

今度は上記の困難さはない。できるかぎり小さい関係に分けるのは本来、module性を良くするためであるから、フレームは、複数個の多項関係で解釈するのが最も良いことになる。次は抵抗器(回路理論の意味で)の解釈である。

[(R-S rs EXT=(#1=(EXT-S a VOLT=va
AMP=ia) #2=(EXT-S b VOLT=vb AMP=ib))
Reg=r) ← va=vb+ia*r .
(R-S rs EXT=(#1=(EXT-S a VOLT=va
AMP=ia) #2=(EXT-S b VOLT=vb AMP=ib))
POWER=p) ← p=ia*(va-vb).
(R-S rs EXT=(#1=(EXT-S a VOLT=va
AMP=ia) #2=(EXT-S b VOLT=vb AMP=ib))
← ia+ib=0 ,(EXT a e).]

但し、小文字で始まる文字列は変数、EXTは外部端子、EXT-Sは外部端子の状態、R-Sは抵抗器の状態に対応する。R-Sが3つの述語に分けてあるのは、module性のためである。goal statement (*)は(**)の略記法と考える。

(*) ← (R-S R1 EXT=(#1=(EXT-S a VOLT=1 AMP=
ia) #2=(EXT-S b VOLT=0 AMP=ib))
REG=1 POWER=1)

(**) ← (R-S R1 EXT=(#1=(EXT-S a VOLT=1
AMP=ia) #2=(EXT-S b VOLT=0 AMP=ib))
REG=1),
(R-S R1 EXT=(#1=(EXT-S a VOLT=1
AMP=ia) #2=(EXT-S b VOLT=0 AMP=ib))
POWER=1).

2.1 データフローによる拘束

多項述語を用いる利点の1つは単一の節を多目的に使用できることである。しかし一般に完全に多目的であることは多くなく、ある程度、変数への入出力関係を制限する必要が生じる。たとえば仕様なしに回路を、回路なしに回路定数を論じることは無意味である。手続の呼出を制限する手段として変数への注釈をよえよう。?が付してある変数は入力変数であり、変数とはマッ

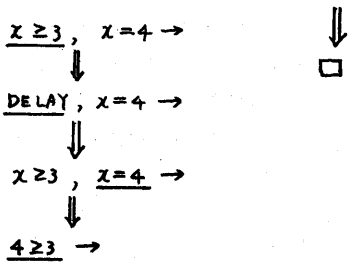
ナしない。^が付してある変数は出力変数であり、変数とだけマッチする。これはIC-PROLOG [clark77] によって提案されているが、ここでは少し機能を拡張して、リテラルの評価順序の変更を行う。(1)はdefault機能(2)はリテラルの評価を遅らせる機能(3)は(2)を利用して手続の多目的化を行った。

(1)リップル含有率が与えられていなければ10%と仮定し、与えてあれば、20%以下であることを確認する。

[(RIPPLE ?rms)←(REAL rms), rms≤0.2.
(RIPPLE 0.1)←.]
但し、REALは、引数が定数であるか否かをテストする述語である。

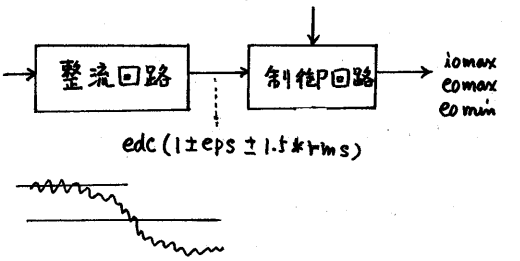
(2)等式は変数が1個以下の場合だけ評価し、もしそうでなければ評価を遅らせる。不等式は変数を含まないときのみ評価し、そうでなければ評価を遅らせる。

[x=y ← (MULTVAR x=y), DELAY.
x=y ← (SOLVEQ x=y).]
[?x≥?y ← (FREEOFVAR x≥y), (GE x-y 0).
x≥y ← DELAY.]
但し、MULTVARは変数が2個以上含まれると真になる述語、FREEOFVARは、変数を含まないとき真になる述語、DELAYは親リテラル(呼出したリテラル)の評価を遅延させるものとする。たとえばgoal statement x≥3, x=4→は次のように導出が進行する。



下線は評価中のリテラルを示す。

(3)直流安定化電源の制御回路の設計。
入力直流電圧edcは、電圧変動率epsで変動し、さらにその上にリップルrms(実効値)がのっている。eomax、eomin、iomax、はそれぞれ最大出力電圧、最小出力電圧、最大出力電流である。edcは、前段の整流回路が与えられているときには与えられ、整流回路の設計も合せて行うときは、与えられない。リップルも同様に必ずしも与えられない。



・[最小入力電圧と最大出力電圧の関係]

最小入力電圧eiminは最大出力電圧eomaxより3v以上高いこと、もし、eiminが与えられなければeomax+3vにとる。

[(EIEO-REL ?eimin ?eomax)←/, (REAL eimin eomax), eimin≥eomax+3.
(EIEO-REL ^eimin ?eomax)←(REAL eomax), eimin=eomax+3.
(EIEO-REL eimin ?eomax)←DELAY.]
[制御回路に必要な定格]
eomin, eimax, edc, eps, rmsから制御回路に必要な定格vcemax(耐圧)、pcmax(電力損失)をきめる。

[(TR-ENV EOMIN=?eomin EOMAX=?eomax EDC=edc EPS=?eps RIPPLE=rms VCEMAX=^vcemax PCMAX=^pcmax)
←(REAL eomin eomax eps),
(RIPPLE rms),

eimax = edc*(1+eps+1.5*rms), ・・・・・①
eimin = edc*(1−eps−1.5*rms), ・・・・・②
(EIEO-REL eimin eomax), ・・・・・③
vcemax = eimax − eomin,
pcmax = vcemax * iomax.]

edc が与えられると①,②,③ の順に評価が進むが、edc が与えられないと、①②は、③の評価が終るまで評価が遅らされる。

[トランジスタ選定]

icmax, vcemax, pcmax, より必要なトランジスタをさがす。検証にも使用できる。

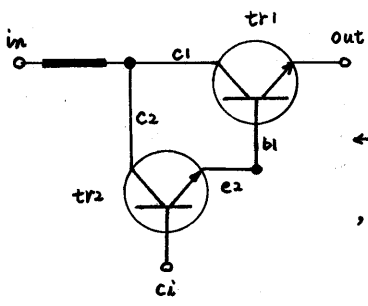
[(TR-SEL ICMAX=?icmax VCEMAX=?vcemax PCMAX=?pcmax HFE=hfe NAME=tr)
← (TR NAME=tr ICMAX=ic pc=pc HFE=hfe VCEO=vceo),
pcmax ≦ 0.8*pc, vcemax ≦ vceo*0.8, icmax ≦ 0.8*ic.]

[制御回路]

仕様を与えて制御回路を設計または検証する。制御トランジスタ一段では、ベース電流が大きすぎるなら駆動トランジスタをかえる。

[(CONTROL SPEC=(EOMAX=?eomax EOMIN=?eomin IOMAX=?iomax EDC=edc
EPS=?eps RIPPLE=rms)
CIR=(NET EXT=(#I=c #O=e #CI=b)
ELE=(TR NAME=tr EXT=(#E=e #B=b #C=c))
NODES=NIL))
← (TR_ENV EOMAX=eomax EOMIN=eomin EDC=edc EPS=eps RIPPLE=rms
VCEMAX=vcemax PCMAX=pcmax),
(TR_SEL NAME=tr VCEMAX=vcemax PCMAX=pcmax ICMAX=iomax HFE=hfe),
(PNP tr), iomax/hfe ≦ 0.1, (EXT c b e).

(CONTROL SPEC=(EOMAX=?eomax EOMIN=?eomin IOMAX=?iomax EDC=edc
EPS=?eps RIPPLE=rms)
CIR=(NET EXT=(#I=in #O=out #CI=ci)
ELE=(TR NAME=tr1 EXT=(#E=out #B=b1 #C=c1))
•(TR NAME=tr2 EXT=(#E=e2 #B=ci #C=c2))
•(WIRE EXT=(#1=w #2=in))
NODES=(NODE EXT=b1.e2)
•(NODE EXT=c1.c2.w)))
← (TR_ENV EOMAX=eomax EOMIN=eomin EDC=edc
EPS=eps RIPPLE=rms VCEMAX=vcemax PCMAX=pcmax)
, (DARLINTON SPEC=(VCEMAX=vcemax PCMAX=pcmax
ICMAX=iomax)
FINAL=tr1 DRIVER=tr2 HFE=hfe),
iomax/hfe ≦ 0.1, (EXT in out ci b1 c1 e2 c2 w).]



制御回路には他にコンプリメンタリ接続等種々あるが、省略する。

```
[(CONTROL   VGAIN=1.0   CIR=(NET   EXT=(#I=c  #O=e  #CI=b)
                                   ELE=(TR NAME=tr EXT=(#E=e #B=b #C=c))
                                   NODES=NIL)) ← .
 (CONTROL   VGAIN=1.0   CIR= 前項に同じ ) ← .]
```

[ダーリントン回路]

```
[(DARLINGTON    SPEC=(VCEMAX=?vcemax  PCMAX=?pcmax  ICMAX=?icmax)
                FINAL=tr1  DRIVER=tr2)
    ← (TR-SEL   NAME=tr1  VCEMAX=vcemax  PCMAX=pcmax  ICMAX=icmax
                HFE=hfe1),(PNP tr1),
                icmax1=icmax/hfe1, pcmax1=pcmax/hfe1,
      (TR-SEL   NAME=tr2  VCEMAX=vcemax  PCMAX=pcmax1  ICMAX=icmax1
                HFE=hfe2), (PNP tr2).]

[(DARLINGTON    FINAL=tr1  DRIVER=tr2
                CIR=(NET  EXT=(#E=e  #B=b  #C=c)
                          ELE=(TR NAME=tr1  EXT=(#E=e  #B=b1 #C=c1))
                              .(TR NAME=tr2  EXT=(#E=e2 #B=b  #C=c2))
                              .(WIRE  EXT=(#1=w  #2=c))
                          NODES=(NODE EXT=b1.e1).(NODE EXT=c1.c2.w)))
                ← (EXT e bc b1 c1 e2 c2 w).]
[(DARLINGTON    FINAL=?tr1  DRIVER=?tr2  HFE=hfe)
                ← (TR NAME=tr1 HFE=hfe1), (TR NAME=tr2  HFE=hfe2), hfe=hfe1*hfe2]
[(DARLINGTON    VBE=1.4) ← .]
[(DARLINGTON    FINAL=?tr1  DRIVER=?tr2  VCEO=vceo) ← (TR NAME=tr1 VCEO=vceo1),
                (TR NAME=tr2  VCEO=vceo2), vceo=(MIN vceo1 vceo2).]
[(DARLINGTON    FINAL=?tr1  DRIVER=?tr2  ICMAX=ic) ← (TR NAME=tr1 ICMAX=ic1
                HFE=hfe1), (TR NAME=tr2  ICMAX=ic2), ic=(MIN ic1 ic2*hfe1).]
                以下略す。
```

2.2 求値問題と手続的解釈

論理プログラミングは証明システムを求値問題にも使えることを提案したものと考えられる。しかし一意解が得られない求値問題は、一般にどの解が最初に求まるべきかを考慮に入れてプログラムしなければならず、手続的解釈が重要な意味を持ってくる。PROLOGでは、これをリテラルと節の記述順序に意味を持たせることによって解決している。しかし、これは手続の多目的性を損う原因になることが多い。たとえば一変数の高次方程式は数値的には解くことができるが、二変数以上では解析的に解かなければならなくなり困難であるが、リテラルの評価順序を変更すれば容易に解決できる場合がある。本稿ではデータフローによるリテラルの実行順序変更によってこの点を解決した。即ち、節はリテラルの記述順序によらない宣言的意味を持たせることができる。

2.3 設計問題に対する接近法

現在までに提案されているいくつかの接近法は次の3つである。

(1) 生成と検証……仕様が満足されるまでデバイスを次々と数えあげる。

(2) 逐次的詳細化……部分的変換によって逐次デバイスの構造を決定していく。

(3) 問題解決法……設計プランの展開と実行による。

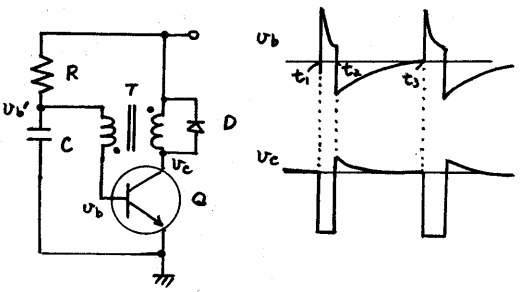
筆者は設計問題のすべて近似は抽象集合を解くことであると仮定した。しかし人間の設計者は対象領域に依存した知識を用いて能率良くこの問題を解決している。このことは抽象の解法を適切に表現する手段が必要なことを示している。LUSHによる方法は上記(1),(2)は容易に表現できる他、メタ述語の考えを使用すれば抽象の解法も自然に表現できる。従ってこの方法は比較的手順が明確になっている分野の設計知識の記述には適当であると考えられる。一方(3)の問題解決法は設計やデバイスの変更を行う行為も対象に含んでいて、設計プランの展開と実行によって問題を解くことを意図している。NASL [Medomott78] はこの例である。電子回路ではこのような構成的理解を利用した設計は比較的多いが、電子回路の理解には使えても、設計に使えるほど充分に形式化できるかどうか疑問の多い所である。

3. 電子回路の理解と説明

1. で述べたように設計者の知識を3つに分けたが、ここでは電子回路の理解システムを考えるうえでの問題点を例をあげてふれることにしよう。

【等価回路による説明】 例としてツェナダイオードによる定電圧回路をとる。(図1.2)は素子定数、回路定数決定のための等価回路である。これを用いて(図1.1)の r_1, r_2 および回路定数を導くことができる。これには回路を解析的に解く必要があるが、線型回路の場合には容易に解くことができる。また近似式の導出も可能である。この方向の理解能力は一般に記号処理の問題となる。

【定性的動作の説明】 次はブロッキング発振器である。この動作は次のように説明される。



[t_1] 正帰還により Q が飽和する。

v_b が V_{BE0} をこえる。
↓
ベース電流 i_b が増す。
↓
コレクタ電流 i_c が増す。
↓
コレクタ電位 v_c が下がる。
↓
ベース電位が上がる。

(t_1, t_2) ベース電流 i_b により C が充電され v_b が降下する。

[t_2] 正帰還により Q がカット・オフされる。

v_b 下がる。
↓
i_b が減る。
↓
i_c が減る。
↓
v_c が上がる。

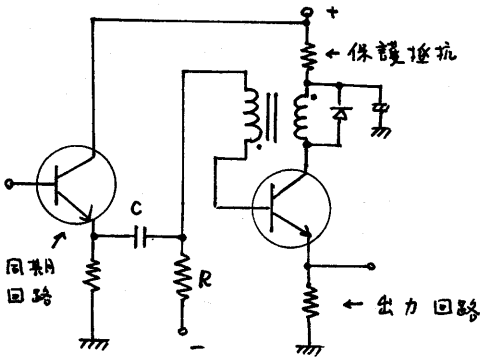
(t_2, t_3) C の電荷が R を通して放電される。

[t_3] t_1 に同じ。

この定性的説明は回路の動作を要約記述したものであるが、その正当性を検証できるシステムが必要である。定

性的因果律の形式化 [kleer 79][Rieger 77] 近似的等価回路への置換による説明システム等の開発が必要であろう。

[構成的記述] 回路の構成過程を記述することにより回路の意味をとえることである。これは回路の性能を査定したり，附加的機能をかんたりするために行われ、従って設計と密接な関係がある。次はブロッキング発振器に同期回路、出力回路、トランジスタ保護抵抗を加えた例である。同期、出力、保護が追加された機能である。



4. おわりに

本稿では、電子回路の設計知識をどのように整理記述し、形式化するべきかについて検討した。拘束による記述法は設計や検証に直接使用でき、現時点では最も実用的であろう。このため他の分野でも使用可能な言語として開発中であり，別の機会に報告する予定である。電子回路理解システムを構成するのが最終目標であるが、解析的理解や構成的理解には、まだ非常に多くの問題を含んでいる。

有益な助言をいただいた本学大型計算機センター、松尾文碩氏に感謝する。

[Kowalski 74] Kowalski RA. (1974)
"Predicate logic as programming languages" IFIP 74.

[Kleer 79] Kleer J. (1979)
"The origin and resolution of ambiguities in causal argument" 6-IJCAI

[Hayes 77] Hayes P.J. (1977)
"In defence of logic" 5-IJCAI

[Charniak 81] Charniak E. (1981)
"A common representation for problem solving and language-Comprehension information"
in Artificial Intelligence Vol 16 No 3.

[Deliyani 79] Deliyani A, Kowalski R.A. (1979)
"Logic and Semantic networks" ACM Vol 22 No 3.

[Hill 74] Hill, R. (1974)
"LUSH resolution and its Completeness." DCL Memo No. 78, Univ of Edinburgh.

[Warren 77] Warren, D.H.D, Pereira L.M, Pereira F.
"PROLOG - The language and its implementation compared with LISP" SIGART newsletters No 64.

[清水] (1977)
"安定化電源回路の設計" CQ出版 k.k

[Mcdermott 78] Mcdermott D. (1978)
"Planning and Acting" in Cognitive Science Vol 2, No 2.

[Mcdermott 78] Mcdermott D.
"Circuit design as problem solving" in AI in CAD North-Holland pub. co. (1978)

[Rieger 77] Rieger, C., Grinberg, M.
"The declarative representation and procedural simulation of causality in physical mechanisms" 5-IJCAI