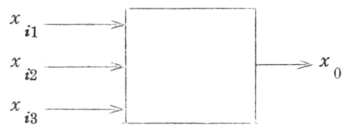
B9　論理シミュレータへのLPの応用

権藤弥栄（日本アイ・ビー・エム東京第一データセンター）

1　論理素子のLPによる表現

ANDや$\overline{\text{OR}}$のような論理素子は，ある種のBlack Boxと考えられ，このBlack Boxの部分をどう作るかによって種々の論理シミュレータが考えられる．

ここでは，このBlack Boxの部分をLPで行うことを考える．



素子のinput，outputを各々x_i，x_0で表わし，次の不等式について考察してみる．

$$x_0 \geq x_i \qquad (1)$$

今，$x_i = 1$とすると，$x_0 \geq 1$となり，ここで，xは，論理素子の入出力値であることから，どんな場合でも，ゼロまたは1の値しかとらないと仮定すると，$x_0 = 1$となる．したがって，不等式(1)は，inputに1を与えるとoutputに1を出す機能を持っていることになる．論理素子は，通常，複数個の入力を持っているので，次に，入力が2個の場合を考えてみる．

$$\begin{cases} x_0 \geq x_{i1} & (2) \\ x_0 \geq x_{i2} & (3) \end{cases}$$

この連立不等式は，次のような機能を持っている．

ⅰ）　$x_{i1} = x_{i2} = 1$のとき$x_0 = 1$

ⅱ）　x_{i1}，x_{i2}のどちらか一方が1のとき$x_0 = 1$

これは，$\overline{\text{OR}}$素子の機能の一部を表現している．$x_{i1} = x_{i2} = 0$のときには，(2)，(3)の連立不等式からは，x_0の値は決定できない．

連立不等式(2)，(3)を$\overline{\text{OR}}$素子の表現として完成させるには，x_0の値が一意的に決定できない場合には“小さい方の値をとる”という規則を作ればよい．この“小さい方の値をとる”という規則は，LPの最小化問題として表現することができる．

次のLP問題を考えてみよう．

$$\overline{\text{OR}}\text{BJ} = x_0 \rightarrow \text{Min} \qquad (4)$$

$$\begin{cases} x_0 \geq x_{i1} & (5) \\ x_0 \geq x_{i2} & (6) \end{cases}$$

ⅰ）　$x_{i1} = x_{i2} = 1$

ⅱ）　x_{i1}，x_{i2}のどちらか一方が1

以上の場合には，制約式(5)，(6)から$x_0 = 1$が一意的に決定し，

ⅲ）　$x_{i1} = x_{i2} = 0$のときには，

目的関数(4)の最小化により$x_0 = 0$となる．したがって，このLP問題は，$\overline{\text{OR}}$機能のすべてを表現している．

以上のような考察を$\overline{\text{NOR}}$，AND，NANDについて行うと，各々次のような最小化ＬＰ問題に帰着できることがわかる．

AND素子

$$\overline{\text{O}}\text{BJ} = -x_0 \qquad (7)$$

$$\begin{cases} x_0 \leq x_{i1} & (8) \\ x_0 \leq x_{i2} & (9) \end{cases}$$

NAND素子

$$\overline{\text{O}}\text{BJ} = x_0 \qquad (10)$$

$$\begin{cases} 1 - x_0 \leq x_{i1} & (11) \\ 1 - x_0 \leq x_{i2} & (12) \end{cases}$$

$\overline{\text{NOR}}$素子

$$\overline{\text{O}}\text{BJ} = -x_0 \qquad (13)$$

$$\begin{cases} 1 - x_0 \geq x_{i1} & (14) \\ 1 - x_0 \geq x_{i2} & (15) \end{cases}$$

目的関数は，最小化問題になるように，AND，$\overline{\text{NOR}}$の場合には，係数にマイナスを付けている．

$\overline{\text{NOT}}$素子は，$x_0 = 1 - x_i$でも表わせるが$\overline{\text{NOR}}$またはNANDでもよいので，ここでは特に取上げなかった．

上述の場合は，ｉｎｐｕｔが２個の場合を考えたが，３個以上でもまったく同じで，ｉｎｐｕｔの数だけ制約式が増えるだけである．

２　Ｆｕｌｌ　Ａｄｄｅｒのモデル化

Ｆｕｌｌ　Ａｄｄｅｒの論理図を図１に示す．これを１章の方式でＬＰモデル化したものが表２である．表２中の目的関数の係数は，次の規則で作られている．

ⅰ）　素子にはｉｎｐｕｔに近いものから順にｌｅｖｅｌを付ける．素子４，５はｌｅｖｅｌ１，素子６，７，８はｌｅｖｅｌ２・・・，以下素子１２がｌｅｖｅｌ６

ⅱ）　素子j　ｏｕｔｐｕｔ変数をx_j，その目的関数中での係数をc_jとすると，c_jは素子jより低いｌｅｖｅｌに属する係数の和より大きくする．

ⅲ）　例えば，素子１２は，一番ｉｎｐｕｔに遠いので，係数を１とし，以下素子１１は２，素子９は４，素子６，７，８は，８，素子４，５は３２と決める．こうして，すべての素子の係数は，その素子より低いｌｅｖｅｌの素子の係数の総和に１を加えたものとなっている．

このように目的関数の係数を決めると，ｉｎｐｕｔに近い素子の出力が優先的に決定されてゆ

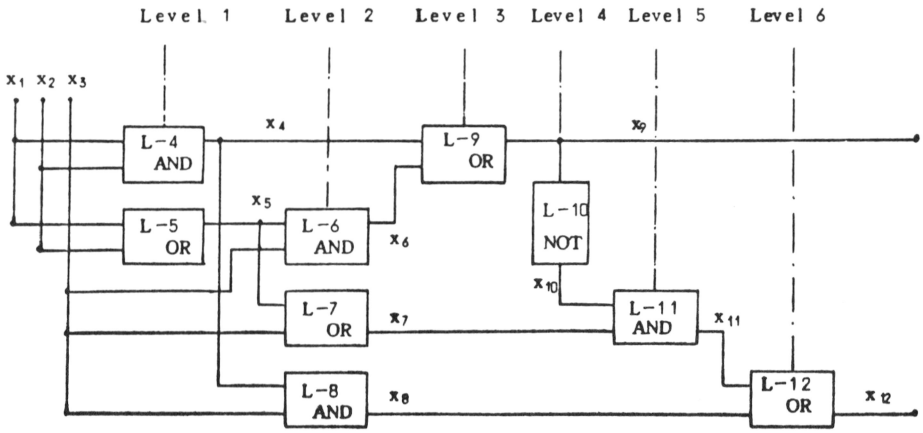


図1　Full adder

			X X X X X X X X X X X X X X X X 1 2 3 4 5 6 7 8 9 0 1 2 1 2 3 4 5 6 7 8
目的関数	OBJ	N	-B B-A A-A A -A 1
L－4	A0	L	-1 1
	A1	L	-1 1
L－5	B0	G	-1 1
	B1	G	-1 1
L－6	C0	L	-1 1
	C1	L	-1 1
L－7	D0	G	-1 1
	D1	G	-1 1
L－8	E0	L	-1 1
	E1	L	-1 1
L－9	F0	G	-1 1
	F1	G	-1 1
L－10	G0	E	1 1 1 1 1 1 1 1
L－11	H0	L	-1 1
	H1	L	-1 1
L－12	I0	G	-1 1
	I1	G	-1 1
入力	INPUT1	E	1 1 1 1
	INPUT2	E	1 1 1 1
	INPUT3	E	1 1 1 1

表2　Full adderのLPモデル

くので，ＬＰモデルを解いた場合，optimal状態では，各論理素子のモデルは必ずその表現すべき素子の機能を果していることになる．

例えば，素子4のinputが$x_1=x_2=1$である場合，x_4の値は制約からは決らず，iterationの過程で決定される．今，$x_4=0$の値をとっているとして，この状態のままoptimalに達することがあり得るだろうか．

$x_4=0$の状態を$x_4=1$の状態に変えると，x_6～x_{12}の値が変化する可能性があるが，その変化が目的関数へ及ぼす影響は，高々x_6～x_{12}までの係数の和にすぎない．この場合には，31を越えることはあり得ない．ところがx_4の係数は，32にしてあるので，$x_4=1$にした方が目的関数の値は小さくなる（x_4の係数は，マイナスとなっている）．したがって，$x_4=0$のままoptimalに達することはあり得ない．

このような事がすべての素子について成立するように係数を決定しているので，ＬＰモデルがoptimalに達すれば，1個1個の素子のモデルは対応する論理機能を果していると言える．

出力は，optimal状態の変数値として決定されるが，入力は，ＬＰモデルの中にはじめから含めておかねばならない．この入力を組込む方法は，LPを解くアルゴリズムに何を使うかによっても異ってくるが，おおよそ，次の2通りがある．

ⅰ） 等式型の制約として入れる．表2がこの方法によるもので，入力値の変更は，RHS Vectorをとり替ることによって行う．

ⅱ） Bound Vectorとして入れるものでこの場合には，アルゴリズムがBound Vectorを扱えるものでなければならない．値の変更は，Bound Vectorをとり替ることによって行う．

この他，入力値を変数として扱わず，定数として扱う方法も考えられるが，入力値の変更がめんどうとなる．

3　ＬＰモデルの特徴

論理をモデル化したＬＰモデルは，次のような特徴を持っている．

ⅰ） 係数行列は，1，0，－1以外の要素を持っていない．

ⅱ） 変数は，0，1の整数値しかとれないから，一見すると，整数条件が必要なようにみえるが，モデルの性質上，整数条件を付けなくても，変数は，整数値をとる．

ⅲ） 変数に0，1のBoundを付けなくても2以上あるいは，－1以下をとることはない．これは，変数に制約が付く場合には，$0\leq x\leq+\infty$，$-\infty\leq x\leq 1$のどちらかの型をしており，前者の場合には$x=0$，後者の場合には，$x=1$となるように目的関数が組立てられているからである．

ⅰ），ⅱ）は，掃出しのどの時点でも，マトリックスの要素が，1，0，－1以外の値をとらないという性質によるもので，このため，ＬＰの解法が簡単化され，演算速度が通常のＬＰよりも早くなる．本論文では，ＬＰを解く部分は，以上のような特徴を考慮した特殊なアルゴ

リズムは，使わず，一般化された，改訂シンプレックス法を使ったが，演算速度を問題にする場合には，以上の特徴を考慮したプログラムを作る必要がある．

4　2桁加算器の演算例

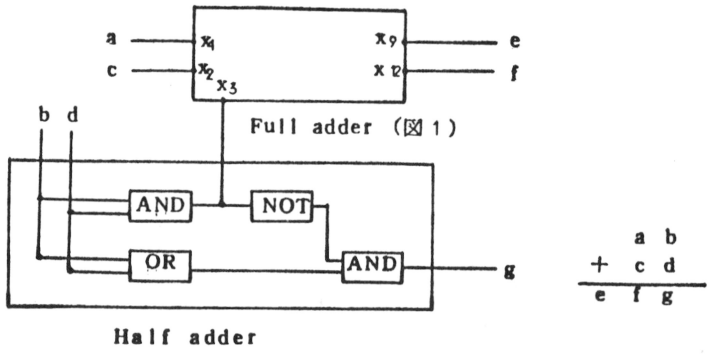


図2.　2桁加算器

図2のようなFull AdderとHalf Adderを結んだ2桁の加算器のシミュレーションを行ってみた．LPのコードには，MPS(LP)を使った．

inputは，制約式にはせずに，Bound Vectorで与えた．MPSのコントロールステートメントは表2に示す．Bound Vectorを使用しているので，1case毎に'SETUP'を行う必要がある．更に，あるinputから次のinputへ移る前に前回の解を'RESTŌRE'する必要がある．

図3は，ōptimalを得るのに要したiteration数である．第1回目は，'CRASH'よりはじめているので16回を要し，以後は，平均4～5回で終っている．Pattern 4とPattern 12では，iterationを要せず，前回のBasisのままoptimalとなっている．このように，あるpatternの結果を出発値にすると，一般に演算効率が向上することが予想される．

inputを制約式としてモデル化した場合には，RHS Vectorをとり替えながら演算を行なうが，この場合には，MPSでは，'SETUP'，'RESTŌRE'

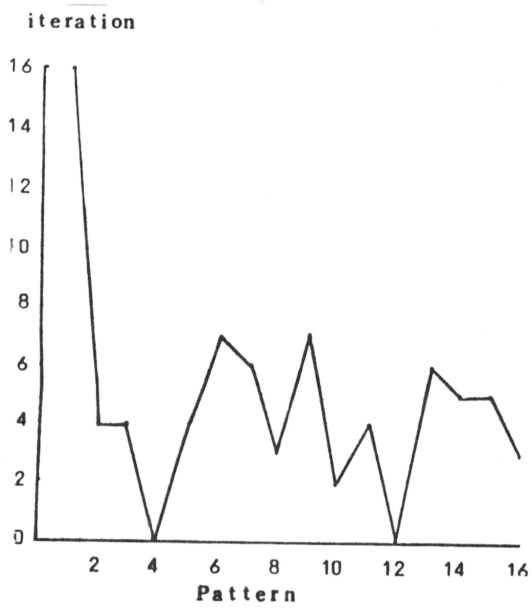


図　3

は不要で，そのまま'PRiMAL'
をかけると，前のBasisがはいっている．この方法は，Bound Vector法より，Procedureが少くてよいが，モデルのRowの数は多くなる．

5 結果と考察

2桁加算器の演算はinput patternのすべての組合せについて行った．結果は，全部で16通りだから，直接調べることができるが，個々の素子が正しく機能しているかどうかは，LP解から知ることができる．

次のようなORの素子を考えてみる．

$$\begin{cases} x_0 \geq x_{i1} & (18) \\ x_0 \geq x_{i2} & (19) \end{cases}$$

これは，LP Matrix中では，Slack Variableが加えられて次のようになる．

$$\begin{cases} x_{i1} - x_0 + S_1 = 0 & (20) \\ x_{i2} - x_0 + S_2 = 0 & (21) \end{cases}$$

ここでx_0が一意的に決定される場合，すなわち，$x_{i1} = x_{i2} = 1$あるいは，どちらか一方が1の場合を考えてみると，その時のSlackは$S_1 = S_2 = 0$，あるいはどちらか一方がゼロとなる．$x_{i1} = x_{i2} = 0$で制約式からだけでは，x_0が一意的に決定されない場合には，itrationの過程で，x_0は，より小さい値をとり$x_0 = 0$となる．この時Slack $S_1 = S_2 = 0$となる．

したがって，Slackの論理積S_1, S_2がゼロとなっている時，個々の素子は正しく機能していることになる．

こうして，LP解のSlack Variableの値を調べることによって，モデルが正しく機能しているかどうかを知ることができる．

以上のような方法で，組合せ回路の論理はLPモデル化でき，それを解くことによって，シミュレーションを完了することができる．

論理シミュレータは，与えられたinput patternに対応するoutput patternを出力するわけだが，このLPモデル化したシミュレータは，output patternを固定することができる．表3は，Full Adderモデルでinputとoutputを，それも論理的に誤ったoutputを固定した例である．この場合には，最適解におけるSlack Variableの論理積がゼロとならない素子が現われ，入出力が矛盾していることを示す．

逆に考えると，論理積がゼロとならない素子を別の素子に変えると入出力は矛盾しなくなることを意味している．

	NUMBER	...ROW..	AT	...ACTIVITY...	SLACK ACTIVITY
	1	OBJ	BS	38.00000	38.00000-
L-4	2	A0	UL	.	.
	3	A1	BS	1.00000-	1.00000
L-5	4	B0	BS	1.00000	1.00000-
	5	B1	LL	.	.
L-6	6	C0	BS	1.00000-	1.00000
	7	C1	BS	1.00000-	1.00000
L-7	8	D0	BS	.	.
	9	D1	LL	.	.
L-8	10	E0	BS	1.00000-	1.00000
	11	E1	UL	.	.
L-9	12	F0	BS	.	.
	13	F1	LL	.	.
L-10	14	G0	EQ	1.00000	.
L-11	15	H0	BS	.	.
	16	H1	UL	.	.
L-12	17	I0	BS	1.00000	1.00000-
	18	I1	BS	.	.
	19	INPUT1	EQ	.	.
	20	INPUT2	EQ	1.00000	.
A	21	INPUT3	EQ	1.00000	.
	22	INPUT9	EQ	.	.
	23	INPUT12	EQ	1.00000	.

入力　$x_1 = 0$, $x_2 = 1$, $x_3 = 1$

出力　$x_9 = 0$, $x_{12} = 1$

（正しい出力は, $x_9 = 1$, $x_{12} = 0$）

表3　矛盾する入力，出力を与えた場合の最適解

本 PDF ファイルは 1972 年発行の「第 13 回プログラミング―シンポジウム報告集」をスキャンし、項目ごとに整理して、情報処理学会電子図書館「情報学広場」に掲載するものです。

この出版物は情報処理学会への著作権譲渡がなされていませんが、情報処理学会公式 Web サイトの https://www.ipsj.or.jp/topics/Past_reports.html に下記「過去のプログラミング・シンポジウム報告集の利用許諾について」を掲載して、権利者の捜索をおこないました。そのうえで同意をいただいたもの、お申し出のなかったものを掲載しています。

過去のプログラミング・シンポジウム報告集の利用許諾について

情報処理学会発行の出版物著作権は平成 12 年から情報処理学会著作権規程に従い、学会に帰属することになっています。

プログラミング・シンポジウムの報告集は、情報処理学会と設立の事情が異なるため、この改訂がシンポジウム内部で徹底しておらず、情報処理学会の他の出版物が情報学広場 (=情報処理学会電子図書館) で公開されているにも拘らず、古い報告集には公開されていないものが少からずありました。

プログラミング・シンポジウムは昭和 59 年に情報処理学会の一部門になりましたが、それ以前の報告集も含め、この度学会の他の出版物と同様の扱いにしたいと考えます。過去のすべての報告集の論文について、著作権者（論文を執筆された故人の相続人）を探し出して利用許諾に関する同意を頂くことは困難ですので、一定期間の権利者捜索の努力をしたうえで、著作権者が見つからない場合も論文を情報学広場に掲載させていただきたいと思います。その後、著作権者が発見され、情報学広場への掲載の継続に同意が得られなかった場合には、当該論文については、掲載を停止致します。

この措置にご意見のある方は、プログラミング・シンポジウムの辻尚史運営委員長 (tsuji@math.s.chiba-u.ac.jp) までお申し出ください。

加えて、著作権者について情報をお持ちの方は事務局まで情報をお寄せくださいますようお願い申し上げます。

期間：2020 年 12 月 18 日 〜 2021 年 3 月 19 日
掲載日：2020 年 12 月 18 日

プログラミング・シンポジウム委員会

情報処理学会著作権規程
https://www.ipsj.or.jp/copyright/ronbun/copyright.html