計算機アーキテクチャ　115－23
設　計　自　動　化　78－23
（１９９５．　１２．　１５）

不完全指定有限状態機械に対する高速な状態数簡単化アルゴリズム

樋口 博之　松永 裕介
(株) 富士通研究所 CAD 研究部
〒211 川崎市中原区上小田中 1015

順序回路の合成において、仕様として与えられる不完全指定有限状態機械の状態数を最小化することは最も重要なステップのうちの一つである。本稿では従来法よりも高速に、よい近似解を求めるヒューリスティックを提案する。提案手法は、全ての極大両立集合の集合を初期解として、逐次改善的に解中の両立集合を縮小および拡大していくことにより与えられた有限状態機械の状態数を削減していくものである。実験を行ったところ、小規模機械に対しては従来よりも高速に全てのベンチマークに対し厳密最小解を求めることができ、大規模機械に対しては計算時間が従来法の 1/20 程度になった。

A Fast State Reduction Algorithm for Incompletely Specified Finite State Machines

Hiroyuki HIGUCHI and Yusuke MATSUNAGA
CAD Laboratory, Fujitsu Laboratories Ltd.
1015, Kamikodanaka, Nakahara-ku, Kawasaki, Kanagawa, 211, Japan

Reducing the number of states in incompletely specified finite state machines(FSMs) is an important step in FSM synthesis. This paper proposes a state reduction algorithm for incompletely specified FSMs. The algorithm utilizes the set of all the maximal compatibles as an initial solution and attempts to improve it iteratively by reducing and expanding compatibles in it. Experimental results are given to demonstrate that the algorithm described here is significantly faster and obtains better solutions than conventional methods.

1 はじめに

順序回路の合成において、仕様として与えられる有限状態機械(FSM)の状態数最小化は最も重要なステップのうちの一つである。FSMの状態数を最小化することにより、より少ないフリップフロップ(FF)で順序回路を実現できる場合がある。また、FFの数が減少しない場合でも、回路の未使用状態が増えることを利用して組合せ回路部分をより簡単化できる。

不完全指定FSMの状態数最小化問題については古くから研究が行われてきたが[1, 3]、NP困難な問題であるため、大規模FSMに対しては実際的な時間で厳密解を得ることは難しい。そこで高速に最小解に近い解を求める、不完全指定FSMの簡単化方法が求められている。以下、不完全指定FSMを単にFSMとよぶ。

従来、FSMの簡単化の方法としては[2, 3]などが提案されている。[2]で提案されている方法は、高速に解を求めることができるが、求められた解の大きさが最小解の2倍程度にもなる場合があった。一方[3]では、状態数(どの状態とも両立的でない状態は除く)が20程度のFSMに対しては実際的な時間で最小解に近い解を求めることができるが、大規模なFSMに対しては膨大な計算時間を要する。そこで、本稿では大規模なFSMにも適用可能な、高速に最小解に近い解を求める方法を提案する。提案手法は、全ての極大両立集合の集合を初期解として、逐次改善的に解中の両立集合を縮小していくことにより与えられたFSMの状態数を削減していくものである。ただし、極大両立集合の数は、最悪の場合与えられたFSMの状態数の指数倍になりうる。そこで本手法では、極大両立集合が多くなりそうな場合には、組合せ的爆発を避けるため、二分決定グラフ(BDD)により非明示的に極大両立集合を表現している。実験を行ったところ、小規模FSMに対しては従来よりも高速に全てのベンチマークに対し厳密最小解を求めることができ、大規模FSMに対しては計算時間が従来法の1/20程度になった。

2 準備

2.1 諸定義

定義 1 初期状態s_aの機械Mに、ある入力系列を印加して遷移する状態系列において、最終状態以外は全て指定された(*don't-care*でない)状態しか現れないとき、そのときのみ、その入力系列は機械Mの状態s_aに対し**印加可能(applicable)**であるという。□

状態s_aに関して印加可能な全ての入力系列の集合をΓ_{s_a}で表す。また、出力関数の定義域を入力系列$\tilde{x}$にも拡張し、入力系列$\tilde{x}$を印加した時の出力系列を$\lambda(\tilde{x}, s_a)$により表すものとする。

定義 2 機械Mの出力系列$\tilde{z}_a, \tilde{z}_b$において対応する確定した出力同士で異なるものがないとき、そのときのみ$\tilde{z}_a$と$\tilde{z}_b$は**両立的**であるといい、$\tilde{z}_a \sim \tilde{z}_b$と書く。□

定義 3 機械Mの二つの状態s_a, s_bに関し、
$\forall \tilde{x} \in \Gamma_{s_a} \cap \Gamma_{s_b}, \lambda(\tilde{x}, s_a) \sim \lambda(\tilde{x}, s_b)$のとき、かつそのときのみ$s_a$と$s_b$は**両立的(compatible)**であるといい、$s_a \sim s_b$と書く。□

定理 1 状態s_a, s_bが以下の二つの条件を満たすとき、かつそのときのみ、それらの状態は両立的である。

1. $\forall x \in I,\ \lambda(x, s_a) \sim \lambda(x, s_b)$
2. $\forall x \in I,\ \delta(x, s_a) \sim \delta(x, s_b)$ □

定義 4 ある状態集合に含まれるどの二状態をとっても両立的であるとき、その集合は**両立的**であるという。□

FSMの簡単化問題の解は、簡単化したFSMの各状態がもとのFSMのどの状態を併合したものかを表す両立集合の集合により表すことができる。以下、簡単化の解と両立集合の集合を同一視する。簡単化の解が満たさねばならない条件は、まず「与えられたFSMの全ての状態を被覆する」ことである。すなわち、もとの状態各々を含む両立集合が少なくとも一つ解に含まれていることである。この条件を**被覆条件**と呼ぶ。

定義 5 両立集合cの入力iに対し、$\bigcup_{s \in c} \delta(s, i)$なる状態集合を**含意集合**という。□

簡単化したしたFSMにおいて、ある状態の、ある入力に対する次状態は一つでなければならないので、両立集合cが解として選ばれると、その次状態集合である各含意集合内の状態は全て一つの状態に併合されなければならない。すなわち、cの各含意集合を含む両立集合が少なくとも一つ解に含まれなければならない。この条件を**閉包条件**と呼ぶ。

定義 6 両立対(s_a, s_b)に対し、次状態対$(\delta(s_a, i), \delta(s_b, i))$を、$i$に対する$(s_a, s_b)$の**含意対**という。□

定義 7 両立集合cに対する**クラス集合**$\mathcal{CS}(c)$は以下を満足する全ての含意集合dからなる集合である。

1. $|d| > 1$、かつ
2. $d \not\subseteq c$、かつ
3. $\forall d' \in \mathcal{CS}(c), d \not\subset d'$. □

定義 8 他の両立集合に含まれない両立集合を**極大両立集合**と呼ぶ。□

2.2 従来法

FSMの簡単化問題は被覆条件と閉包条件を同時に満たす、できるだけ要素数の少ない両立集合の集合を求める問題である。しかし、この2つの条件を同時に考慮することは困難である。そこでKannanら[2]は、まず被覆条件のみ考えてできるだけ小さな両立集合の集合を作り、その後閉包条件を満たすように両立集合を追加する、FSMREDという方法を提案した。

一方、Rhoら[3]は、STAMINAというFSMの状態数最小化システムを開発し、その中で3つのヒュー

リスティックを提案している。これらは、以下のような方法に基づくものである。まず全ての極大両立集合を生成し、それらの両立集合のみを考えて、最小な被覆閉包集合をまず求める。次に、その最小集合に含まれるプライムな両立集合を全て計算し、最小集合にそれらのプライムな両立集合を加えた両立集合のみを考えて、(厳密に) 最小な被覆閉包集合を求める。

FSMRED は、一般に STAMINA よりも高速だが、簡単化能力の点では劣っている。また、STAMINA の方は、全ての (プライムな) 両立集合を考慮して厳密に最小被覆閉包問題を解くことは膨大な計算量を要するので、極大両立集合のみ考え、それに対して最小被覆閉包問題を解いている。しかし、極大両立集合の数も最悪状態数の指数倍になり、大規模 FSM に対しては膨大な計算時間を要するという問題点があった。

3 不完全指定 FSM の簡単化

本章では、不完全指定 FSM の簡単化のための、高速なヒューリスティックを提案する。

3.1 概要

本手法は、初期解に対し、被覆閉包条件を満たしつつ解中の両立集合の縮小と拡大を繰り返しながら、解中の不要な両立集合を削除して FSM を簡単化する。

被覆条件と閉包条件を満足する初期解を計算するため、以下の定理を用いる。

定理 2 全ての極大両立集合の集合は、解の一つである、すなわち被覆条件と閉包条件を満足する。 □

定理 2 から、初期解として全ての極大両立集合からなる集合を用いることができる。提案手法 SLIM (SequentiaL machIne Minimizer) の概略を以下に示す。

[FSM 簡単化のためのヒューリスティック]

1. マージ表を作る
2. 解 $S \leftarrow$ (全ての極大両立集合)
3. $S \leftarrow$ REDUCE(S)
4. $S \leftarrow$ MERGE(S)
5. $S \leftarrow$ EXPAND(S)
6. $S \leftarrow$ REDUCE-II(S)
7. $S \leftarrow$ MERGE(S) □

SLIM では、まずマージ表 (merger table)[4] を作り、全ての状態対についてその両立性を調べる。次に、マージ表をもとに全ての極大両立集合を生成し、それを初期解とする。極大両立集合の数は最悪の場合状態数の指数倍になり、大規模 FSM に対しては明示的に生成できない場合がある。その場合には、BDD を用いて非明示的に生成する。この BDD は両立集合の辞書のようなものとして用い、それ以降の処理に必要なものだけ明示的に取り出して処理する。

	NS, z			
PS	I_1	I_2	I_3	I_4
A	$B, 0$	$-, -$	$-, -$	$E, 1$
B	$A, 0$	$C, -$	$-, -$	$-, -$
C	$C, -$	$A, -$	$D, 1$	$E, -$
D	$-, -$	$B, -$	$-, -$	$A, -$
E	$C, -$	$B, 1$	$F, -$	$-, 0$
F	$F, 1$	$A, -$	$E, -$	$B, -$

図 1: FSM M

初期解が求まると、被覆条件と閉包条件を満たしつつできる限り、解が変化しなくなるまで繰り返し各両立集合を縮小していく REDECE という処理を行う。処理の途中で、他の両立集合に被覆された両立集合は削除する。その後、MERGE では解 S の中で、二つ以上の両立集合が一つにまとめられる場合に一つに併合する。続いて、先の REDUCE により得られた極小解から、さらによい解を求めるため、極小解から抜け出す手続き EXPAND を行う。EXPAND では、解 S 中の各両立集合を閉包条件が満たされている限りできるだけ拡大する。次によりよい解を見つけるために、先ほどの REDUCE による縮小とは異なる方法で両立集合の縮小を行い、最後に再び MERGE を行う。

本手法では両立集合に優先度をつけその順に両立集合を処理することが多い。両立集合は、その時点でまだ被覆されていない状態をできるだけ多く含むほど、またそのクラス集合が小さいほど最小解に含まれる可能性が高いので、そのような両立集合の優先度が高くなるような尺度を導入している。

3.2 REDUCE

REDUCE は、初期解として与えられる極大両立集合全てからなる集合に対して行われる。被覆条件のみを考えると極大両立集合よりよい両立集合は存在しない。しかし、閉包条件を考えると極大両立集合を解に含めるより、それに含まれる、より小さな両立集合を用いる方が含意集合が小さくなり閉包条件が緩くなる場合がある。従って、解中の両立集合を縮小することにより、不要な両立集合が解の中に存在するようになり、より小さな解が求められる。REDUCE では、現在の解の閉包条件を満たしてできるだけ各両立集合を縮小する。例えば、図 1のような FSM M を考える。M の初期解と各極大両立集合のクラス集合は図 2のとおりである。 この初期解において、両立集合 DEF を考えると、その閉包条件を満足するには、解中の両立集合 ABC および CEF それぞれに含まれる状態のうち、部分集合 AB および CF のみが必要である。このようにして、全ての両立集合に対してその閉包条件を満足するために必要な部分を抽出すると、解として閉

両立集合	クラス集合
$\{A,B,C\}$	$\emptyset$
$\{B,C,E\}$	$\{A,B,C\},\{D,F\}$
$\{B,D,E\}$	$\{A,C\},\{B,C\}$
$\{C,E,F\}$	$\{A,B\},\{D,E,F\},\{B,E\}$
$\{D,E,F\}$	$\{A,B\},\{C,F\}$

図 2: 初期解

Reduce($CompSet\ S$) {
1. **while** (S が更新された) {
2. **for** ($c \in S$) $core(c) \leftarrow \emptyset$;
3. $Q \leftarrow S$ における必須両立集合の集合; $U \rightarrow \emptyset$;
4. **for** ($c \in Q$) $core(c) \leftarrow c$ 内の必須状態の集合;
5. **if** ($Q = \emptyset$) {
6. $Q \leftarrow core(c) = \emptyset$ なる c のうち優先度最大のもの c_m;
7. c_m の核集合を c_m 自身にする;
8. }
9. **while** ($Q \neq \emptyset \vee U \neq \emptyset$) {
10. $c(\in Q)$ をひとつ選び Q から取り除く; $U \leftarrow U \cup c$;
11. **for** (c の class set 内の各要素 d について) {
12. $d \subseteq c'(\in S)$ なる c' を選ぶ;
13. **if** ($core(c') = \emptyset$) $Q \leftarrow Q \cup \{c'\}$;
14. $core(c') \leftarrow core(c') \cup d$;
15. }
16. }
17. 核集合の集合が被覆・閉包条件を満たしているか調べる;
18. 他の核集合に含まれている核集合を除く;
19. $S \leftarrow$ 核集合の集合;
20. }
}

図 3: REDUCE の処理の流れ

包条件を満足するのに必要な部分を求めることができる。このように REDUCE では、解の各両立集合中で解となるために必要な部分集合を抜き出す処理を逐次改善的に解に変化がなくなるまで繰り返す。その処理の流れを図 3に示す。図 3の 3 行目に現れる必須両立集合は以下のように定義される。

定義 9 簡単化の解 S に関して、ある両立集合 c が他のどの両立集合にも含まれていない状態を含んでいるとき、c は S における**必須両立集合**であるという。 □

10 行目から 18 行目で閉包条件を満足するのに必要な各両立集合の部分集合を計算する。この集合を各両立集合に対する核集合とよぶ。核集合の計算は、解 S 内の両立集合を順に処理し、その閉包条件を満足するよう核集合を拡大していくことにより行う。その際、全ての両立集合を処理するまでに被覆閉包集合が求まる場合がある。そのときは、それ以降の両立集合を処理する必要がない。処理する必要のある両立集合の数は、処理する両立集合の順に依存するが、必須両立集

両立集合	クラス集合	含意集合に対する c'	核集合
ABC	$\emptyset$	–	ABC
DEF	AB, CF	ABC, CEF	DEF
CEF	AB, DEF, BE	ABC, DEF, BDE	CF
BDE	AC, BC	ABC, ABC	BE
BCE	ABC, DF		$\emptyset$

図 4: REDUCE における核集合の計算

合は被覆条件を満足するためには必ず処理する必要がある。そこで 3 行目でまず必須両立集合を求め、存在する場合にはそれから最初に処理する。それ以外の場合には優先度最大の両立集合から処理を行う。14 行目では、両立集合 c のクラス集合の要素 d が含まれるべき両立集合 c' を一つ求め、その核集合に d の状態を加える。解の大きさは c' の決め方にも依存するが、できるだけ核集合が拡大しないよう c' を決定する。

図 2の初期解に対して図 3の while ループの処理を 1 回行ったときの核集合は図 4のとおりである。各欄は、それぞれ左から解中の両立集合、そのクラス集合、クラス集合内の含意集合に対する c'(図 3)、および各両立集合に対して計算された核集合を示す。この図で、両立集合の順は処理された順を表し、BCE は処理する必要がなかったことを表す。また、必須両立集合は ABC のみであり、その必須な状態は A である。この処理により、例えば両立集合 CEF は CF に縮小していることが分かる。CF のクラス集合は $\{BE, DE\}$ であり、CEF のそれより小さいため、次の縮小でさらに解が縮小される可能性があることが分かる。

3.3 MERGE

REDUCE 後の解中には、複数の両立集合を併合して一つにしてもなお閉包条件を満たすものが存在する。MERGE では解中の任意の両立集合の組に対し併合できるかどうかを調べ、併合可能なら併合を行う。

3.4 EXPAND

REDUCE は逐次改善的手法であるため、局所解に陥りやすい。そのため、MERGE を行った後、解中の各両立集合を拡大して局所解から抜け出す処理を EXPAND では行う。具体的には、解中の各両立集合がそれを含むある極大両立集合に拡大しても閉包条件を満足するときには拡大を行う。

3.5 REDUCE-II

EXPAND により解中の各両立集合を拡大した後、REDUCE で行った両立集合の縮小とは異なった方法で両立集合の縮小を行う。REDUCE では、核集合を計算する際各両立集合自体のクラス集合を用いてい

た。しかし、実際には各両立集合のうち必要なのはその部分集合である核集合だけである。従って、そのクラス集合も両立集合自体のクラス集合よりも小さい場合があるが、REDUCEではその点が考慮されていない。そこでREDUCE-IIでは、両立集合単位で閉包条件を満足するための計算を行うのではなく、それより小さな両立対の単位で処理を行う。REDUCE-IIを行う解はすでにそれ以前の処理により十分解が小さくなっていると考えられ、REDUCEより綿密な計算を行っても高速性は保たれる。

4 実験結果

前章までに述べた手法に基づくFSMの簡単化プログラムSLIMをC++言語を用いて実現した。Sun-4/10上でMCNCベンチマークに対するFSMの簡単化の実験結果を表1に、いくつかの大規模FSM[5]のうちSTAMINAの近似解法で簡単化できるFSMに対する実験結果の平均を表2に、STAMINAで簡単化できないFSMに対する実験結果を表3にそれぞれ示す。表において、欄"状態数"はFSMの状態数を示す。"orig"はもとのFSMの状態数、"min"は厳密に最小化したFSMの状態数を表す。欄"CPU時間"は簡単化に要したCPU時間を示す。STAMINAはrubin600–2250に対してはマージ表を作った時点で両立対がないという間違った解析をして終了したため、結果が得られなかった。本実験では、両立集合を明示的に生成するか非明示的に生成するかの両立対の数のしきい値は10000とした。提案手法SLIM(表中"slim")に対する比較としてFSMRED("fr")[2]およびSTAMINAの近似解法("stam")[3]による結果も同時に示す。ただし、FSMREDの大規模FSMに対する結果は得られていない。STAMINAの近似解法は3種類あるが、そのうち各FSMについて状態数の最小のもの、全て同じ場合は実行時間の最短のものを選択した。

表1から提案手法は全てのMCNCベンチマークに対して最小解が得られた。表1に現れないFSMは両立対が存在しないか全ての状態が1つに併合されたかのいずれかである。STAMINAの近似解法も表1では全て最小解を得ているが、提案手法の平均約8倍も計算時間を要している。FSMREDはscfを除いて高速であるが、状態数が多い。一方厳密最小化については、ex2の場合計算時間が2400秒以上も必要であった。

次に大規模FSMに対する実験結果である表2と表3について考察する。従来法がいくつかのFSMに対し結果が出せなかったのに対し、提案手法は実験を行った全ての大規模FSMに対して、簡単化を行うことができた。状態数については、提案手法は従来法に対して、平均87%程度ですんでおり簡単化の能力が従来法より高いことがわかる。例題の中には、提案手法を行ったときの状態数が、STAMINAを行ったときより半分程度になった場合もあった(th55)。最小解と比較しても、最小解が得られているほとんどのFSMにおいて、提案手法でも最小解が得られている。最小解が得られていないFSMについても、今までで最も小さな解が求まっている。またCPU時間については、従来法のSTAMINAと比較して1/20程度と極めて高速であった。この値はSTAMINAでは結果が出せなかった表3のFSMを除いて平均をとったものであることを考慮すると、提案手法と従来法の差はさらに広がる。

提案手法がこのように高速であったのは、提案手法が厳密に最小被覆閉包問題を解いていないためであると考えられる。STAMINAでは、3つのヒューリスティックとも、解空間をしぼりこんでから最終的には最小被覆閉包問題を解いている。MCNCベンチマークのような小規模FSMに対しては、解空間をしぼりこむことにより解空間のサイズが極めて小さくなり、最小被覆閉包問題を解いてもそれほど多くの計算時間を要しないが、表3のように大規模になると膨大な計算時間を要してしまう。FSMの簡単化問題では被覆条件だけでなく閉包条件も考慮する必要があるため、解中に最小解には含まれない両立集合を1つ含めることによりその閉包条件を満足するために多くの不要な両立集合も解に含めなければならないことが多い。従来法のFSMREDはまず被覆条件を考えてから、後から閉包条件を考える方法であったが、FSMREDの状態数が他の方法に比べて多くなってしまうのはまさにその点に原因があった。これは厳密に最小被覆閉包問題を解かない場合には、ヒューリスティックを注意して作らないと状態数が増大することを示している。その点で提案手法はほぼ最小解に近い解を高速に見つけることができており、簡単化に有効な手法であるといえる。

本手法はいくつかの手続きを組み合わせて一つのヒューリスティックになっている。本実験において各処理が終った時点での状態数を表4に示す。表中の数は左からそれぞれ与えられたFSMの状態数の総和、REDUCE後の状態数の総和、1回目のMERGE後の状態数の総和、簡単化が終った後の状態数の総和、最小状態数の総和である。ただし、rubin600–2250の状態数は元の状態数が他に比べて膨大なため、総和には含めていない。ただし、最小解が得られていない例題については、本実験も含めて今までで最も小さな解を最小解として用いた。この表から、REDUCEにより1/4程度に状態数が削減できていることが分かる。それに続くMERGEでものべ30もの状態が削減でき、この処理の有効性が示されている。REDUCE-IIによる状態の削減はわずかに2であるが、これはこの前の段階でほとんど最小解が得られているためであると考えられる。しかし、REDUCE-IIにより状態数が1つ

表 1: 実験結果 (MCNC ベンチマーク)

FSMs	状態数				CPU 時間 (秒)		
	orig/min	slim	fr	stam	slim	fr	stam
bbara	10/7	7	7	7	0.03	0.07	0.02
bbsse	16/13	13	–	13	0.10	–	0.03
beecount	7/4	4	4	4	0.02	0.03	0.00
ex1	20/18	18	–	18	0.13	–	0.04
ex2	19/5	5	10	5	0.12	0.10	15.47
ex3	10/4	4	–	4	0.06	–	0.15
ex5	9/3	3	5	3	0.07	0.03	0.05
ex7	10/3	3	4	3	0.05	0.03	0.07
lion9	9/4	4	–	4	0.06	–	0.01
mark1	15/12	12	–	12	0.13	–	0.02
opus	10/9	9	–	9	0.02	–	0.00
scf	121/97	97	97	97	0.34	33.27	0.41
sse	16/13	13	13	13	0.09	0.03	0.02
tbk	32/16	16	16	16	0.94	1.17	1.09
train11	11/4	4	4	4	0.08	0.03	0.02

表 2: 実験結果の平均 (大規模 FSM)

方法	平均状態数		平均時間 (秒)
	orig	結果	
SLIM	36.48	5.26	0.96
STAMINA	36.48	6.05	18.61

表 3: 実験結果 (STAMINA で結果を得られなかった FSM)

FSMs	状態数		時間 (秒)
	orig	slim	slim
ifsm2	48	9	0.20
rubin600	600	3	12.43
rubin1200	1200	3	51.53
rubin2250	2250	3	204.06

ずつ削減された 2 つの FSM では、それにより厳密な最小解が得られている。また、REDUCE-II の前に行われる EXPAND は局所最適解から抜け出すためのものであるが、例えば表 1 の ex7 では、EXPAND を行わずに REDUCE-II を行っても最小解は得られない。また最初に REDUCE のかわりに REDUCE-II を行うことも考えられるが、その時点ではまだ対象となる両立集合の数が多いため、処理時間も多く必要とする。また REDUCE-II を行うためには 2 つ以上必須な状態を持つ両立集合を探し、必須な状態の対から処理を行うが、両立集合が多い場合には、そのような対が存在せず全く縮小できない場合も多いと考えられる。これらのことから EXPAND の処理も必要不可欠であるといえる。

提案手法では、両立的な状態対がある与えられた値よりも大きい場合には、両立集合を非明示的に処理する。実験ではそのしきい値を 10000 としたところ、rubin600–2250 に対して非明示的に両立集合が生成された。これらの回路は状態数がそれぞれ 600,1200,2250 であり、厳密最小解を求める際に考慮しなくてはならない両立集合であるプライムな両立集合の数が状態数の指数倍になるように人工的に作られた FSM である。これらの FSM は実用的なものではないが、本手法は BDD を用いることにより、このような大規模 FSM に対しても 3,4 分程度で簡単化することができている。

5 おわりに

不完全指定 FSM の簡単化手法 SLIM について述べた。本手法は実験を行った結果、全ての MCNC ベンチマークについて最小解を得ることができ、従来法よりも高速であった。大規模 FSM に対しても、計算時間が従来法の 1/20 程度になり本手法の有効性が示された。

表 4: SLIM の各処理後での状態数 (総和)

	orig.	reduce の後	1 回目の merge 後	結果	最小
状態数	1227	371	347	345	342

参考文献

[1] A. Grasselli and F. Luccio. "A Method for Minimizing the Number of Internal States in Incompletely Specified Sequential Networks". *IRE Trans. on Electronic Computers*, 14(3):350–359, June 1965.

[2] L. N. Kannan and D. Sarma. "Fast Heuristic Algorithms for Finite State Machine Minimization". In *Proceedings of European Design Automation Conference*, pages 192–196, February 1991.

[3] J.-K. Rho, G. Hachtel, F. Somenzi, and R. Jacoby. "Exact and Heuristic Algorithms for the Minimization of Incompletely Specified State Machines". *IEEE Trans. on Computer-Aided Design*, 13(2):167–177, February 1994.

[4] Z. Kohavi. *Switching and Finite Automata Theory 2/e.* Tata McGraw-Hill, 1978.

[5] T. Kam, T. Villa, R. Brayton, and A.Sangiovanni-Vincentelli. "A Fully Implicit Algorithm for Exact State Minimization". In *31st ACM/IEEE Design Automation Conference*, pages 684–690, June 1994.