モデル検査器を用いた自動検針システムの仕様検証

早水　公二[†]　　篠崎　孝一[††]
高橋　孝一[†]　　渡邊　宏[†]

仕様書の査読検証に代わる新しい検証手段を確立するため、モデル検査器を用いて仕様を検証する事例研究を行った。本研究のために作成した自動検針システムの仕様書を題材に、有限状態遷移系を構成し、並行して仕様書中の各仕様を時相論理 CTL(Computation Tree Logic)の論理式で表現し、モデル検査を行った。検査ツールは SMV(Symbolic Model Verifier) を用いた。検査の結果、仕様書中の矛盾や記述不足の不具合を発見することができた。発見された不具合は仕様書作成時の査読では考慮しきれなかったものである。本事例研究を通してモデル検査器を用いた仕様の検証の実行可能性を示す。

†　独立行政法人　産業技術総合研究所　システム検証研究ラボ
††　関西電力株式会社　総合技術研究所

目次

１．はじめに

利用者の要求を満たすシステムを設計・製作するためには、システムへの要求を静的な状態及び動的な振る舞いとして定義した仕様書が必要である。システムの設計・製作は仕様書に記述された状態及び振る舞いをＨ／ＷとＳ／Ｗを用いて実現することによって利用者の要求を満たすことである。仕様書の内容に矛盾、間違い、記述不足等の不具合があると、製作したシステムにも不具合が生じて、利用者の要求を満たすことができず、改修・再製作が必要となる。しかし、一旦製作したシステムを改修、再製作することは極力避けるべきである。なぜなら、仕様書の修正作業とは比較にならないほど多くの時間と労力を必要とするからである。実際に企業等では、改修・再製作にかかるコストが、システムの設計・製作のコストの中で大きな比重を占めている。従ってシステムの計画から製作に至る過程の早い段階で、理想的には仕様書作成時に、潜在する不具合を見つけ出し取り除くことが大事である。

我々は通常、平易な自然言葉で簡潔に仕様を記述するが、システムが取り得る全ての状態、振る舞いを完全に網羅して記述することは容易ではない。相容れない二つ以上の動作が同時に起こる不具合や、記述不足等の不具合が仕様に混入することは珍しいことではない。そこで仕様の査読検証を行うが、発生頻度の低い状況下で起こる潜在的な不具合は簡単に発見できない。なぜなら、査読検証では、査読者の持つ経験や知識に頼る部分が大きく、知識や経験から外れる状況を想定することは困難であり、不具合を見過ごしやすいからである。そこで査読者・経験・知識に依存しない仕様の検証方法、すなわち第三者による機械的な検証方法が必要となる。今回、これを実現する一つの試みとしてモデル検査技術を利用し仕様の検証[SY]を行った。

モデル検査[HR、CGP、B+]では、システムを有限個の状態を持つモデルで表現し、システムが満たすべき性質がモデル上で成り立つかどうかを検査する。検査は全てのパス上で性質が成り立つかどうかを機械的・網羅的に行う。しらみつぶしに検査するため、極めて特別な状態で性質が成り立たない場合も必ず発見することが可能である。査読による検証方法では特別な状態は発見しにくい。モデル検査は 80 年代後半から研究が盛んになって来た比較的新しい技術で、ハードウェア検証の分野では既に実用化の域にある。モデル検査器は計算機上で自動的にモデル検査を行うツールであり、90 年代に入ってから種々のモデル検査器が発表されている。

本研究では、モデル検査器を用いてシステム仕様の検証を行った。検証ではシステムの仕様書からモデルを作成し、仕様書中の各仕様それぞれをシステムが満たすべき性質として検査する。仕様の検証結果は、個々の検査の結果から評価される。モデルは性質を満たすはずであるので、全ての検査に合格すれば仕様の検証が完了したことになる。もし検査に合格しなければ、仕様書中に不具合があると考えられる。特にモデル検査器の中には、検査に合格しない場合に反例を出力するものがあり、反例は不具合個所の特定に役立つはずである。図 1 に本研究の手順を示す。

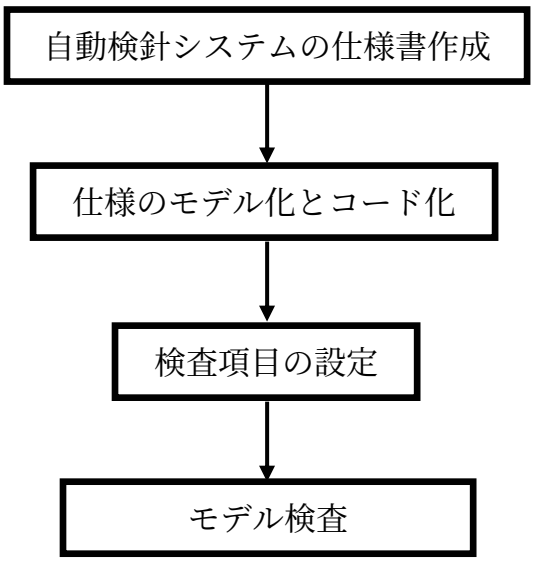

図 1　研究の手順

今回は事例として、電力自動検針システムを取り上げる。自動検針システムとは、センタからの遠隔操作によって、ゲートウェイ（以下、GW（GateWay））を介して電力量計（以下、WHM（Watt Hour Meter））のメータ値の読み取り、データの設定を行うシステムである。WHM内でのデータ設定と、センタ～GW～WHM間の通信が中核的な処理となる。まず、設備の動作仕様及び設備間の通信仕様をまとめてシステム仕様書とした。作成した仕様はモデル化を行った後、今回用いたモデル検査器であるＳMVで用意されたモデル記述用の言語を用いてコード化した。検査項目は、システム仕様書に記述した各設備の動作仕様とシステム全体の動作に関する項目とし、モデル検査器ＳMVでの検査項目記述用の言語となっ

ているＣＴＬ（Computation Tree Logic）の時相論理式（以下、ＣＴＬ論理式）で表現し、仕様のコードと併せてＳＭＶへ入力し検査を行った。

本研究では、以上のような方法で自動検針システムのシステム仕様の検証を行い、「２つの仕様の衝突」や「仕様の記述不足」など仕様書作成時には考慮しきれなかったいくつかの不具合を発見し、それを取り除くことができた。これにより、実システムの仕様検証への一歩を踏み出すことができた。

以下、本稿では、モデル検査の概略と今回検証に用いたＳＭＶと検査項目記述用の言語ＣＴＬについて説明した後、自動検針システムの仕様、仕様のモデル化とコード化、検査項目の設定及び具体的な検査方法とその結果について順を追って説明する。そして最後に、仕様検証の結果、検証の過程で得られた知見等を考察としてまとめる。

謝辞

本研究の一部は文部科学省科学技術振興調整費（若手任期付研究員支援）による成果である。

２．モデル検査と時相論理式

２．１　モデル検査

２．１．１　モデル

モデルとは、初期状態を含む有限個の状態と、それらの状態間の遷移から構成される状態遷移系である。状態遷移系を図で表現したものが状態遷移図である。図 2 に状態遷移図の例を示す。

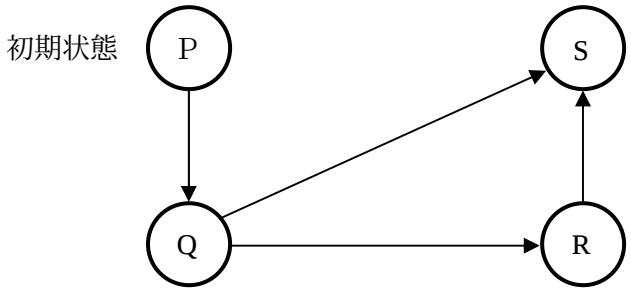

図 2　状態遷移図

図 2 の状態遷移図は、４つの状態（状態Ｐ、状態Ｑ、状態Ｒ、状態Ｓ）と、それぞれの状態間の遷移（４つの矢印：→）で構成される。遷移の矢印で繋がる状態の列（例：Ｐ→Ｑ→Ｒ）を「パス」と呼ぶ。

２．１．２　モデル検査

モデル検査とは、そのモデルが満たすべき性質（そのモデルに要求される仕様＝検査項目）が、入力されたモデルの取り得る全てのパス上で成立するかどうかを、網羅的に検査することである。例えば図 2 で示したモデルに以下の２つの性質１と性質２が要求される場合を見てみよう。

●性質１：
「いかなるパスにおいても、いつかは必ずＳが成立する」
●性質２：
「いかなるパスにおいても、いつかは必ずＲが成立する」

●性質１の検査結果は「正しい（true）」である。

・Ｐ→Ｑ→Ｒ→Ｓのパスで、Ｓが成立する。

・Ｐ→Ｑ→Ｓのパスでも、Ｓが成立する。

性質１は、モデルが取り得る全てのパス上で成立するため、検査結果は「true」となる。

●性質２の検査結果は「間違い（false）」である。

・Ｐ→Ｑ→Ｒ→Ｓのパスでは、Ｒが成立する。

・Ｐ→Ｑ→Ｓのパスでは、Ｒが成立しない。

性質２は、モデルが取り得る全てのパス上で成立するとは言えない。従って、検査結果は「false」となる。

モデル検査器は、以上のようなモデル検査を計算機上で機械的かつ自動的に行う。モデル検査についての詳細は、入門書[HR、CGP、B+]を参考にされたい。またモデル検査を始めとする数理的検査技法一般については形式的技法の Web ページ[FM]にて紹介されている。

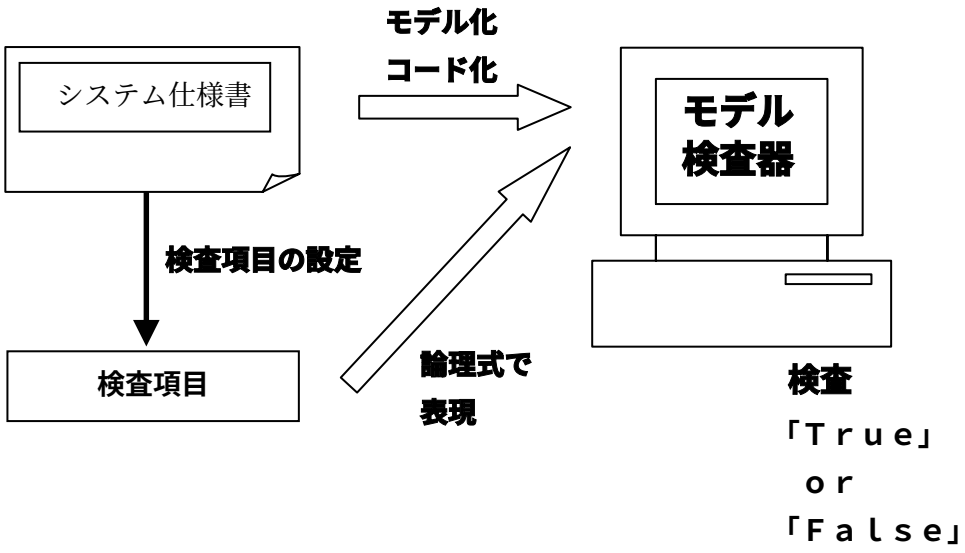

図３　モデル検査器を用いた仕様検証の概略

本研究では，電力自動検針システムの仕様書を事例に、仕様書からモデルを作り、仕様書中の各文章を満たすべき性質として仕様書のモデル検査を行なった。モデル検査器はＳＭＶ[MC、CA]を採用した。モデル検査器ＳＭＶを用いた仕様検証[図３ 参照]では，まず仕様書に記述されたシステムの仕様を状態遷移図にモデル化する。さらにモデル記述用の言語を用いてコード化する。これと並行して、検査項目記述用の言語（ＣＴＬ）を用いて検査したい項目をＣＴＬ論理式で表現する。これらのモデルのコードとＣＴＬ論理式を SMV に入力すると、ＳＭＶはＣＴＬ論理式がモデル上で正しいか間違っているかを検査し、検査結果「true」あるいは「false」を出力する。「false」の場合は同時に反例も出力する。モデル検査結果が「true」の場合は仕様上で検査項目が成り立つことが確認できる。逆に「false」と反例が出力された場合には，仕様上で検査項目が成り立たず仕様に何らかの不具合があることがわかり、さらに反例を調べると仕様の不具合個所が特定できる。

２．２　時相論理式

時相論理は「次に信号が青に変わる」、「将来，電車が来る」や「3 分以内に電源が切れる」などの時間に関わる性質を述べるための言語であり、時相論理式はその言語によって記述された文である。モデル検査では、システムが満たすべき性質を記述するのに時相論理式を使う。「次の瞬間」、「次の次の瞬間」というような飛び飛びの時間しか許さないものや、「3 分 19 秒後」などの連続的な時間を許すものなど、時間の種類によって様々な種類の時相論理が提案されている。また、モデル検査器ごとに採用されている時相論理が異なり、記述できる性質が若干異なる。モデル検査器 ＳＭＶでは，飛び飛びの時間を持つ時相論理ＣＴＬ(Computation Tree Logic)が採用されている。時相論理ＣＴＬ は命題論理[表 1 参照]に時間に関する論理記号を 8 つ[表 2 参照]を加えて拡張したものである。

表 1　命題論理

記号	使用例	意味
＆	φ＆ψ	φとψの論理積（and）
｜	φ｜ψ	φとψの論理和（or）
！	！φ	φの論理否定
－＞	φ－＞ψ	φならばψ　（！φ｜ψと同じ）

表 2　時間に関する論理記号

記号	使用例	意味
ＡＧ	AG(φ)	全てのパスにおいて、常にφが成立する
ＡＸ	AX(φ)	全てのパスにおいて、次にφが成立する
ＡＦ	AF(φ)	全てのパスにおいて、将来のある時点でφが成立する
ＡＵ	A[φUψ]	全てのパスにおいて、ψが成立するまでφが成立する
ＥＧ	EG(φ)	あるパスにおいて、常にφが成立する
ＥＸ	EX(φ)	あるパスにおいて、次にφが成立する
ＥＦ	EF(φ)	あるパスにおいて、将来のある時点でφが成立する
ＥＵ	E[φUψ]	あるパスにおいて、ψが成立するまでφが成立する

注）　A：All　E：Exist
　　G：Globally　X：neXt　F：Furture　U：Until

一般に検査項目は自然言語で記述する。モデル検査器ＳＭＶを使う場合、前もってそれぞれの検査項目をＣＴＬ論理式で表現する必要がある。

自然言語の記述とＣＴＬ論理式の記述の見掛けの違いは大きく、ＣＴＬ論理式を作成する作業は容易ではない場合が多い。しかし作成の際，自然言語で記述された検査項目の文章に、ＣＴＬ特有の記号が意味する「すべてのパスにおいて」、「あるパスにおいて」、「常に」、「次に」、「将来」、「ある時点までは」などの表現を補うと書き換えが容易になる。例えば「φが絶対成り立たない」という検査項目は、「全てのパスにおいて、常にφが成立しない」と書き直して、ＣＴＬ論理式 AG(!φ)(「全てのパスにおいて、常に !φ が成立する」)と表現する。

３．自動検針システムの仕様

自動検針システムの仕様をシステム仕様書[H1]としてとりまとめた。ここでは、自動検針システムの概要を述べる。

３．１　自動検針システムの概要

自動検針システムは「センタ」、「GW」、「WHM」の３つの設備から構成され、センタ～GW間、GW～WHM間にそれぞれ双方向の通信路がある。センタからの遠隔操作によって、GWを介してWHMのメータ値の読み取り、データの設定を行うシステムである。センタはGWに対して要求を出力して、その要求に対する応答としてGWから処理結果の報告を入力する。GWはセンタからの要求に応じて、WHMに対して指令データを出力して、WHMの電力メータ値の読み込み、WHMのパターンデータ設定を行う。WHMはGWからの指令データに応じて電力メータ値の返送、パターンデータ設定の処理を行い、指令データに対する応答としてGWに返送を出力する[図 4 参照]。

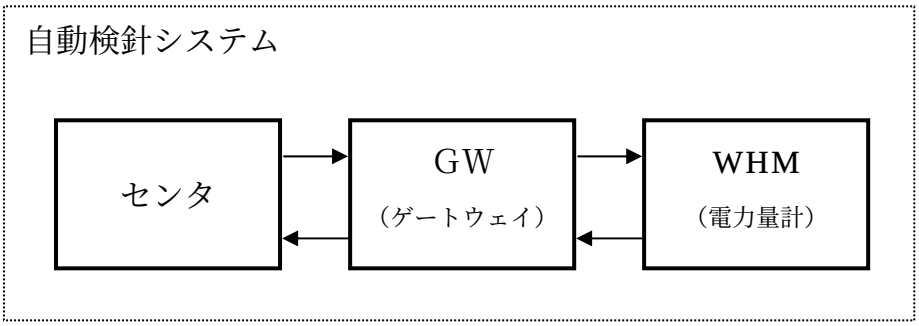

図 4　自動検針システムの構成図

３．２　各設備の動作仕様

各設備の動作仕様をまとめたシステム仕様書[H1]の中のセンタ、GW、WHM の動作仕様の一部をそれぞれ抜粋して付録１に載せる。

４．仕様のモデル化とコード化

仕様のモデル化の作業では、システム仕様書で定義されたシステムの静的な状態及び動的な振る舞いに、変数やその値、条件を割り当て、それらの相互の関係を状態遷移図で表現する。また仕様のコード化では、モデル化によって得られた状態遷移図を、モデル記述用の言語でコードに変換する[H2]。システム仕様のモデル化とコード化の手順を図 5 に示す。以下、それぞれの作業について順を追って説明する。

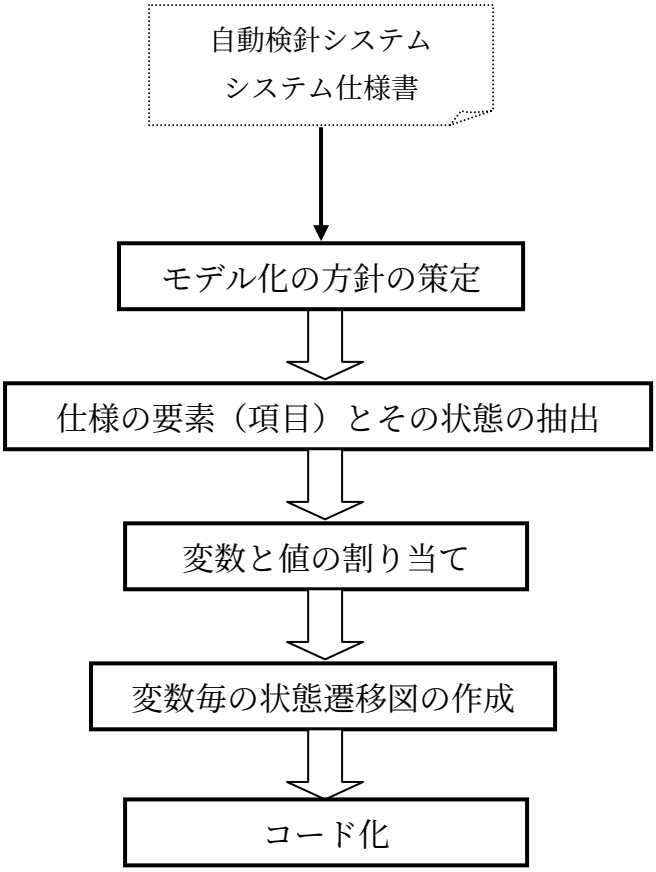

図 5　システム仕様のモデル化とコード化

４．１　モデル化の方針の策定

システム仕様をモデル化する際には、まずモデル化の方針を立てる。「方針を立てる」とはモデル化の対象を明確にすることである。システム仕様の中で検査に直接影響を及ぼすと考えられる内容や、検査したい項目と密接に関係すると考えられる仕様を抽出する。逆に、仕様書には記述されているが検査には直接影響を及ぼさないと考えられる内容、あるいは検査では必要がないと考えられる詳細な仕様を省略する。また、モデル検査器は無限個あるいはそれに近い状態を扱うことはできないため、システムを有限個の状態から成るモデルで表現する。モデル化の対象を明確にし、不要な仕様を省略することにより、モデルが取り得る状態数を少なくできた。自動検針システムの検証では、以下の２つの方針を立てた。

●設備の電源状態を記述する

自動検針システムの検証では、WHM内でのデータ設定と、センタ～GW～WHM間の通信に重点を置いているため、各設備の動作や設備間の通信に直接影響を及ぼす設備の電源状態に着目し、電源の状態変化がシステムに及ぼす影響をモデルに記述する。

●通信内容を抽象化する

設備間でやりとりする通信の内容（電文の構成、内容等）は、データ設定及び通信に直接影響を及ぼさない。従って、モデルでは通信内容は可能な限り省略する。(ただし、GWから出力する指令データの正常／異常はモデルに記述する)

４．２　項目と状態の抽出

次に、立てた方針に従ってシステムを構成する項目とその項目が取り得る状態を抽出する。項目とは仕様書の記述内容の中のキーワードである。自動検針システムの検証では、WHM内でのデータ設定と、センタ～GW～WHM間の通信に重点を置いて仕様書を作成したため、以下の言葉が仕様書中で頻繁に記述されている。

「電源」、「通信」

「要求」、「指令データ」、「返送」

「パターン設定」...

これらの言葉は仕様書のキーワードであるので項目として列挙する。各

項目にはその項目が取り得る状態が存在する。例えば、電源は「正常」と「異常」の２つの状態が存在し、要求は「要求無し」、「要求有り」、「待機状態」の３つの状態が存在する。

抽出された項目と状態は、システムの静的な状態が持つ特徴を端的に捉えるものである。的確な項目及び状態の抽出の成否は、検証の成否に大きな影響を与える。

４．３　変数と値の割り当て

抽出された項目と状態にそれぞれ、変数と値を割り当てる。自動検針システムの場合、次のように項目に変数を割り当てた。

GWの電源（Ｐｏｗｅｒ）は、「GW＿P」

通信（Ｃｏｍｍｕｎｉｃａｔｉｏｎ）は、「Ｃｏｍ」

各状態にも次のように値を割り当てた。

電源の正常／異常は、「ＯＮ」／「ＯＦＦ」

要求の待機状態は、「Ｗｔ」

自動検針システムの項目と状態及びそれぞれに割り当てた変数と値の一覧を付録２に載せる。

４．４　状態遷移図の作成

最後に変数毎に状態遷移図を作成した。状態遷移図はシステムの動的な振る舞いを過不足無く表現するものである。自動検針システムの変数 Rpt の場合は図 6 のように書いた。変数 Rpt は、「センタへの報告」の項目を表す変数[付録２参照]である。これは、報告を何も出力していない「報告無し」の状態と、処理が正常に完了したことを報告する「完了」の状態、処理が正常に完了しなかったことを報告する「失敗」の状態を持ち、それぞれに「EMP」「OK」「NG」の値を割り当ててある。矢印の上の①～④は、遷移が起こる条件であり、条件が満たされたときだけ遷移が起こる。例えば、状態が EMP であるとき条件①が成立するならば、次の瞬間に OK に遷移し、条件②、③、④のうちのいずれかが成立するならば、NG に遷移する。

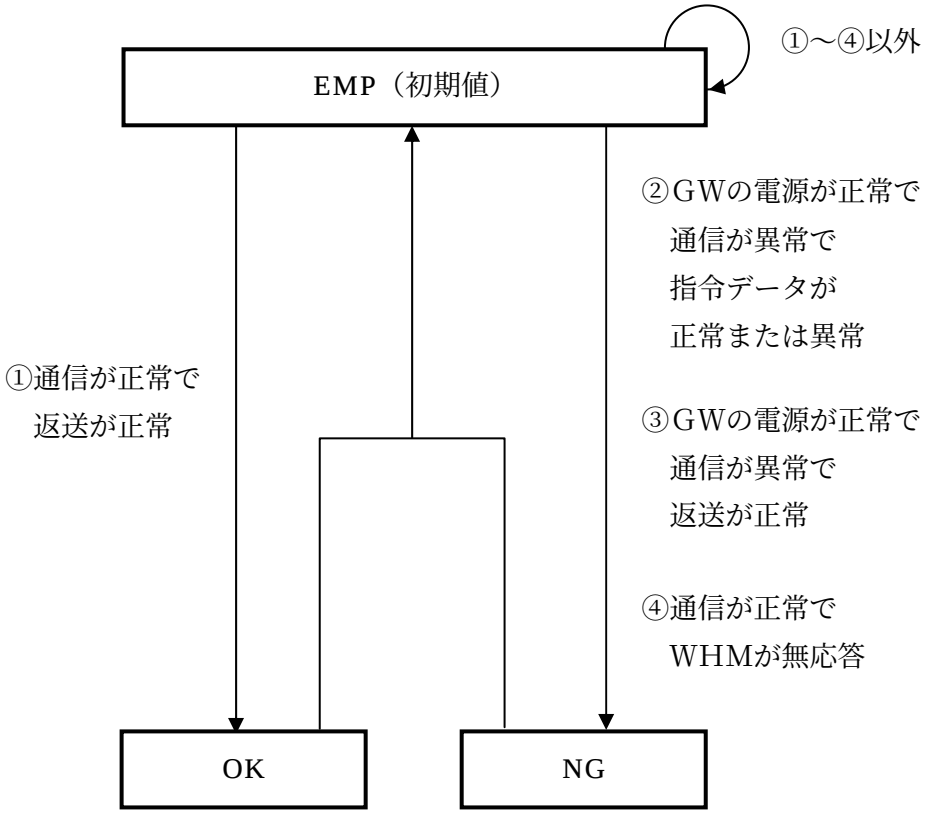

図 6　変数 Rpt の状態遷移図

４．５　仕様のコード化

各変数について作成した状態遷移図を、ＳＭＶのモデル記述用言語でコード化する。図 7 は図 6 の変数 Rpt の変数宣言と状態遷移図のコードである。

```
MODULE  main

VAR
  Rpt : {EMP, OK, NG};

ASSIGN
  init(Rpt) := EMP;
  next(Rpt) := case
                Rpt = EMP & Com = 1 & Ret = OK : OK;
                Rpt = EMP & CGW_P = ON & Com = 0 & Ord_Dt != EMP : NG;
                Rpt = EMP & CGW_P = ON & Com = 0 & Ret = OK : NG;
                Rpt = EMP & CCom = 1 & Ret = NR : NG;
                1 : EMP;
              esac ;
```

図７　自動検針システムの仕様のコード

MODULE　main　：　コードの開始部である。ＳＭＶでは、通常のプログラミング言語と同様、コードは関数形式で表現する。

VAR　：　変数の宣言部である。ここでは、変数 Rpt と、その値 EMP、OK、NG を宣言している。

ASSIGN　：　処理の記述部である。変数の振る舞い（値の変化）を記述する。変数 Rpt は、

・初期値（init）が EMP

・次の値（next）は条件（case 内）による。

１行目～４行目は図６の①～④を示す。

５行目は①～④以外を示す。

５．検査項目の設定

５．１　検査項目の選定

本検証では、自動検針システムのシステム仕様書に記述された各設備の動作仕様の全てと、それぞれの動作仕様を組み合わせることによって得られるシステム全体の動作仕様を検査項目とした。

（1）設備の動作仕様

３章で用意した自動検針システムの仕様書[付録１ 参照]の「GWの動作仕様」と「WHMの動作仕様」の各仕様の項目をそのまま検査項目とした。ただし今回はWHM内でのデータ設定と、センタ～GW～WHM間の通信に重点を置いている。従って、センタからの要求の出力は、GWでの要求の入力処理を検査するための手段として考え、センタ自体の動作仕様は検査項目に含めない。

（2）システム全体の動作仕様

システム全体についての動作仕様として次の（ａ)、（ｂ）２つを検査項目とした。

（ａ）パターン設定完了の確認

センタ～GW～WHM間の通信が正常に動作し、WHM内でのデータ設定も正常に動作すれば、自動検針システムの重要な動作の１つであるパターンデータの設定が完了する。従って、次の検査項目を設けた。

「センタから要求が出力されれば、最終的にWHM内でのパターン設定が完了すること」

（ｂ）初期状態への再帰可能性

自動検針システムはどんな状態からでも、デッドロックにおちいらず、いつかは初期状態に戻ることが期待される。従って次の検査項目を設けた。

「システムが初期状態から動作して、いかなる状態からでも再び初期状態へ戻ることができること」

５．２　検査項目のＣＴＬ論理式での表現

用意した検査項目をＣＴＬ論理式で表現した。以下に検査項目とＣＴＬ論理式の例を３つ挙げる。

（1）例１　「WHMの動作仕様」－（4）－（a）－②

「全てのパターンデータのメモリ蓄積が完了した場合、WHMの電源が正常の場合は、本体へのパターンデータの設定が完了した時点で、メモリは「空データ」で上書きされる」

上の文章をＣＴＬ論理式で表現する。最初の作業として、「２．２　時相論理式」で述べたようにＣＴＬ特有の表現を補って次の文章に書き換えた。

「システムがいかなるパスにおいても、①全てのパターンデータのメモリ蓄積が完了し、かつWHMの電源が正常であるならば、②いかなるパスを選択しても次に必ずメモリは空データで上書きされる」

この文章の各文節にはそれぞれ次のような論理式が部分的に対応する。

「いかなるパスにおいてもφとなる」	AG φ
「全てのパターンデータのメモリ蓄積が完了」	P_Cnt = ALL
「WHMの電源が正常」	WHM_P = ON
「いかなるパスを選択しても次に必ずφとなる」	AX φ
「メモリは空データで上書きされる」	P_Cnt = EMP

従って、下線部①には次のＣＴＬ論理式が対応する。

```
P_Cnt = ALL & WHM_P = ON
```

また、下線部②には次のＣＴＬ論理式が対応する。

```
AX ( P_Cnt = EMP )
```

以上をまとめて、全体を次のＣＴＬ論理式で表現した。

```
AG ( ( P_Cnt = ALL & WHM_P = ON ) -> AX ( P_Cnt = EMP ) )
```

（2）例2　「WHMの動作仕様」－（4）－（b）

「全パターンデータのメモリ蓄積が完了した場合でも、WHMの電源が異常の場合は本体への設定は出来ないこととする」

上の文章をＣＴＬ論理式で表現する。ＣＴＬ特有の表現を補って次の文章に書き換えた。

「システムがいかなるパスにおいても、①全てのパターンデータのメモリ蓄積が完了し、かつ本体へのパターンデータの設定が未完了で、かつWHMの電源が異常であるならば、②いかなるパスを選択しても、WHMの電源が正常となるまでは、本体へのパターンデータの設定が未完了のままである」

この文章の各文節にはそれぞれ次のような論理式が部分的に対応する。

「いかなるパスにおいてもφとなる」	AG φ
「全てのパターンデータのメモリ蓄積が完了」	P_Cnt = ALL
「本体へのパターンデータの設定が未完了」	P_Set = OFF
「WHMの電源が異常」	WHM_P = OFF
「いかなるパスを選択してもψまではφとなる」	A[φUψ]
「WHMの電源が正常」	WHM_P = ON

従って、下線部①には次のＣＴＬ論理式が対応する。

P_Cnt = ALL & P_Set = OFF & WHM_P = OFF

また、下線部②には次のＣＴＬ論理式が対応する。

A[(P_Set = OFF) U (WHM_P = ON)]

以上をまとめて、全体を次のＣＴＬ論理式で表現した。

```
AG ( ( P_Cnt = ALL & P_Set = OFF & WHM_P = OFF ) ->
                    A[ ( P_Set = OFF ) U ( WHM_P = ON ) ] )
```

なお、本検査項目では「WHMの電源が正常となるまでは」とあるので、WHMの電源が異常の状態から、正常の状態へ遷移するパスで検査を行うことが付加条件として必要である。付加条件については、「６．１　検査の際の付加条件」で述べる。

（3）例3　「5．1－（2）システム全体の動作仕様」－（b）

「システムが初期状態から動作して、いかなる状態からでも再び初期状態へ戻ることができる」

上の文章をＣＴＬ論理式で表現する。ＣＴＬ特有の表現を補って次の文章に書き換えた。

「システムがいかなるパスにおいても、あるパスを選択すれば将来のある時点で初期状態に戻る」

この文章の各文節にはそれぞれ次のような論理式が部分的に対応する。

「いかなるパスにおいてもφとなる」　AG φ

「あるパスを選択すれば将来のある時点でφとなる」　EF φ

「初期状態になる」は、「全ての変数が初期値になる」と同義であり、

```
GW_P = ON & WHM_P = ON & Req = OFF & Ord_Dt = EMP
  & Ret = EMP & Rpt = EMP & P_Cnt = EM & P_Set = OFF
  & Req_Hty = EMP & Dt_Hty = EMP
```

以上をまとめて、全体を次のＣＴＬ論理式で表現した。

```
AG ( EF ( GW_P = ON & WHM_P = ON & Req = OFF & Ord_Dt = EMP
  & Ret = EMP & Rpt = EMP & P_Cnt = EMP & P_Set = OFF
  & Req_Hty = EMP & Dt_Hty = EMP ) )
```

６．モデル検査と結果

ここでは、検査の際の付加条件及び、具体的な検査作業とその結果ついて述べる[H3]。

６．１　検査の際の付加条件

ＳＭＶは入力されたＣＴＬ論理式の真偽（true／false）を、モデルの取り得る全てのパス上で機械的かつ網羅的に検査する。ＣＴＬ論理式が成立しないパスが１つでもあると、モデル検査の結果は偽となる。ただし自動検針システムの事例では、ＣＴＬ論理式の中で必ずしも全てのパス上で成り立つことが必要でないものもある。例えば、「５．２　検査項目のＣＴＬ論理式での表現」－（２）例２がそうである。この検査項目は、WHMの電源が異常のままとどまらず、必ず正常な状態へ遷移するパスの上で検査することが付加条件となる。付加条件をＳＭＶに入力するには、論理記号「ならば（->）」の前提条件に含めてしまうことが望ましい。しかしＣＴＬ論理式で表現することが困難な付加条件は、ＳＭＶコードの中にデバッグ文として挿入したり、モデル記述用言語の規約「FAIRNESS 文」を用いる。この例の場合、以下のような「FAIRNESS」で条件を付加する。

```
FAIRNESS　WHM_P = ON
```

上の FAIRNESS 文は、変数 WHM_P が無限回 ON の状態となるパスのみを残し、それ以外のパス、すなわち付加条件を満たさないようなパスをモデルから除外して検査することをＳＭＶに指定する。上の付加条件で除外されるパスを以下に例示する。

① WHM_P = OFF, OFF, OFF, OFF, OFF, OFF, OFF ・・・（以下 OFF）
② WHM_P = OFF, **ON, ON, ON,** OFF, OFF, OFF ・・・（以下 OFF）
③ WHM_P = OFF, **ON,** OFF, OFF, **ON,** OFF, OFF ・・・（以下、OFF）

①、②、③では変数 WHM_P は、高々有限回しか ON の状態にならない。これらのパス上で例２を検査すると偽となる。

６．２　検査作業とその結果

６．２．１　ＳＭＶへのＣＴＬ論理式の入力

ＣＴＬ論理式はモデルのコードの下に追記（ 図 8 参照 ）し、ＳＭＶへ入力する。図 8 では FAIRNESS 文も追記されている。

```
MODULE  main

VAR

ASSIGN                    ＜検査項目の宣言部＞
                          １つの「SPEC」につき１つの
FAIRNESS                  ＣＴＬ論理式を記述する
   WHM_P = ON

SPEC
   AG ( ( P_Cnt = ALL & WHM_P = ON ) -> AX ( P_Cnt = EMP ) )
SPEC
   AG ( ( P_Cnt = ALL & P_Set = OFF & WHM_P = OFF )
            -> A[ ( P_Set = OFF ) U ( WHM_P = ON ) ] )
SPEC

  ・
  ・
  ・
  ・
```

図 8　ＣＴＬ論理式を追記したコード

６．２．２　検査の結果

自動検針システムのモデルとＣＴＬ論理式をＳＭＶに入力し、モデル検査を行った。用意したＣＴＬ論理式の数は約 50 個であり、全てのＣＴＬ論理式を検査した。作業開始当初、「true」が出力されないＣＴＬ論理式がいくつかあった。それらを調べると次のような状況にあることがわかった。

①仕様のモデル化あるいはコード化に誤りがあった。
②検査項目をＣＴＬ論理式で表現する際に誤りがあった。
③仕様書に不具合があった。

①および②に該当する検査項目については，検査作業上発生したミスであるので，モデル化やコード化、ＣＴＬ論理式を見直すことでミスを取り除き、検査結果「true」が出るようにした。５０個のＣＴＬ論理式の内、４８個のＣＴＬ論理式について「true」が出力されることを確認した。しかし、ミスを取り除いても、「true」にならない検査項目が 2 つあった。それは上の ③仕様の不具合が見つかったことを意味する。「５．２　検査項目のＣＴＬ論理式での表現」の例１及び例３がそれである。不具合が出た検査項目 2 つの検査結果についてそれぞれ説明する。

（１）例１の検査結果

モデル検査器ＳＭＶは「false」を出力し、次のような反例のパスを出力した。

反例：全てのパターンデータのメモリ蓄積が完了し、かつ本体へのパターンデータの設定が完了し、かつWHMの電源が正常の状態で、WHMがＧＷからパターン設定指令１データを入力したため、メモリは空データで上書きされず、パターン１のみのデータで上書きされた。

この反例は、仕様書中の次の２つの仕様が関わる。

（３）－⑤
：メモリが「空データ」の状態、あるいは上記②、③、④の後にパターン設定指令１データを入力すると、メモリはパターン１のみのデータで上書きされ、パターン１のデータから蓄積を開始する。

（４）－（ａ）－②
：本体へのパターンデータの設定が完了した時点で、メモリは「空データ」で上書きされる。

反例の理由は次のように分析できる。上の 2 つの仕様の記述は互いに独立したものであり、仕様書の上では同時に起こることが可能である。しか

し、実際には同時に起こってはならないことがらであり、仕様書作成時に見落とされて独立した記述になっていた。状態遷移図作成時には、これに気付かず、2 つの仕様が独立した関係ではなく、仕様（４）－（ａ）－②よりも（３）－⑤が必ず優先されるモデルにしてしまった。その結果、パターン設定が完了したにも関わらずメモリが空データで上書きされず一方の仕様が満たされない反例が生じた。これは 2 つの仕様が両立せず矛盾する不具合の例である.

この仕様の矛盾を解消するために、２つの仕様が同時に動作するような状況では、（３）－⑤の動作仕様が優先されることとした。解消策は、以下のように太字の部分を追記して、システム仕様書に反映した。

（３）－⑤

：メモリが「空データ」の状態、あるいは上記②、③、④の後にパターン設定指令１データを入力すると、メモリはパターン１のみのデータで上書きされ、パターン１のデータから蓄積を開始する。

（ただし空データの上書きと同時に、GWからパターン設定指令１データを入力した場合は、パターン１の処理を選択し、メモリはパターン１のみのデータで上書きすることとする。）

仕様変更後の本検査項目のＣＴＬ論理式は次のように修正した。

```
AG ( ( !( Com = 1 & Ord_Dt = P1 ) & P_Cnt = ALL & WHM_P = ON) -> AX ( P_Cnt
                                                                = EMP ) )
```

（２）例３の検査結果

モデル検査器ＳＭＶは「false」を出力し、次のような反例のパスを出力した。

反例：センタから要求が出力されたが、GWの電源が異常でありその要求を入力出来ない。またGWは電源異常のため、要求を入力できなかった旨を報告としてセンタに出力することができない。そのためセンタはGWからの報告を待ち続ける「待機状態」を維持し続け初期状態である「要求無し」の状態に戻れず、システム全体としても初期状態に戻ることができない。

センタの動作仕様は検査の対象外であったが、この反例によって図らずも、「センタの動作仕様」の次の仕様が記述不足であることがわかった。

（４）　：待機中にGWから報告が出力された場合は、センタとGW間の通信が正常であれば報告を受け付け、待機状態を抜け次の要求をGWに対して出力する。

反例のパスを経ると初期状態に戻ることができないことが発見できた。これは、センタが待機状態のまま“フリーズ”してしまい、システム全体がフリーズするパスである。原因は、センタが待機中にGWから報告が出力されなかった場合の動作仕様を記述していないためである。すなわち仕様書の記述不足が原因であった。これを解消するために、センタが待機状態となった場合は、GWから報告が出力されなくても、タイムアウトにより待機状態を抜けることとした。解消策は、以下のように太字の部分を追記して、システム仕様書に反映した。

（４）　：待機中にGWから報告が出力された場合は、センタとGW間の通信が正常であれば報告を受け付け、待機状態を抜け次の要求をGWに対して出力する。
また、センタは待機状態となった時点でカウントを開始し、GWから報告を入力できずカウンタが「７」となった場合は待機状態を抜けることとする。（これは、待機状態のままフリーズする状態を回避するためである）

６．３　検査のまとめ

当初用意した自動検針システムのシステム仕様書に２つの不具合があることがわかった。それらを改修し、再度検査し、全ての検査項目について「true」が出力されることを確認した。

７．考察

ここでは、本研究の成果を企業等において実務レベルで採用する場合（社内技術として活用する場合も含む）の有用性について考察する。

７．１　研究に用いたシステムとその仕様書

本研究に用いた自動検針システムは、関西電力総合技術研究所にて開発・試作された実験用システムである。モデル検査用に特化した架空のシステムではない。システムの仕様書は、本研究のために特徴的な機能に絞り込んで新たに作成した。仕様書は自動検針システムの実際の動作を参考にして仕様を取り決め文書化したものであり、モデル検査を行いやすいように特別に配慮して作成したものではない。企業等でシステムを構築する際に通常作成するシステム仕様書と変わりはない。

７．２　仕様書の査読の有無

作成した仕様書は、初歩的な不具合を発見するため共同研究者による査読を行った。査読を行ったのは、自動検針システムの開発や情報システムの仕様書作成の業務経験を持つ研究者であり、仕様書を作成する段階で十分なチェックを実施した。

従来からの方法では、この後さらにシステム発注者、設計者、プログラマー等の査読を要する。本研究ではこれらの査読検証の代わりに、モデル検査器を用いた仕様の検証を行った。

７．３　モデル検査器を用いた仕様検証の結果

「6. モデル検査と結果」で述べたように、仕様書に潜在した不具合を２つ発見することができた。仕様書作成時に行った共同研究者による査読では、これら２つの不具合は発見できなかった。

７．４　発見した不具合の特徴と査読による発見の可能性

発見した２つの不具合の特徴を分析する。不具合１は同時あるいは短い時間幅の中で起こる不具合であり、時間的なタイミングまで考慮しなければ発見できない。不具合２は処理を場合分けして記述する際に、全ての場合について記述しているかどうかをチェックしなければ発見できない。

従来からの方法を採用し、査読を行った場合に２つの不具合を発見できていたかどうかを推測する。不具合２は、処理の場合分けを確実に行い、

全ての場合に沿って処理が記述されているかをチェックすれば発見できた可能性が高い。しかし、不具合１は人間が最も苦手とする時間的はタイミングを考慮する必要があり、査読では発見できなかった可能性がある。また、シミュレーションによる試験、あるいは実機による試験を行っても、時間的なタイミングに起因した不具合は一般に発見しにくい。

７．５　従来からの方法との比較

本研究では、従来からの方法であるシステム発注者、設計者、プログラマー等の査読は行っていないため、そこにかかる時間とコストは算出できない。またモデル検査器を用いた仕様検証にかかった正味の時間とコストも算出できない。研究に要した時間とコスト（約４人月）には、自動検針システムの動作を理解し、検査器に習熟し、研究環境の整備をするために必要な時間とコストが含まれているからである。従って、現時点では時間とコストの視点から両者を比較することはできない。

７．５　モデル検査器を用いた仕様検証の有用性

実際に開発・試作されたシステムを用いたこと、通常作成するシステム仕様書の検証ができたこと、業務経験を持つ研究者の査読でも発見できなかった不具合を発見できたこと、発見した不具合の中にはシミュレーション試験や実機試験でも発見しにくい不具合があったこと、これらのことから、本研究で提案する新しい仕様検証の手法は、企業等において実務レベルで採用する場合にも有用であると考えられる。ただし現時点では、時間とコストの面で、従来の方法より優れているとは言えない。従って、本手法は、時間とコストよりも不具合発見を最優先するようなシステムに適用すべきである。このようなシステムは、不具合が発生すれば人命、社会に重大な被害をもたらす大規模システム、実運用で不具合が発生した場合に多大な改修費用を要する大量生産システムを指す。例として、大規模発電プラントの制御システム、携帯電話等が挙げられる。

以上のことを総合して、次のようにまとめる。

検証のために多少の時間とコストの代償を払っても、実運用までに絶対に不具合を発見しなければならないシステムの開発～製作段階で、モデル検査器を用いた仕様検証を適用することは有用である。

８．おわりに

従来からの査読による仕様の検証方法に代わって、モデル検査器を用い

た仕様の検証を行い、企業等において実務レベルで適用する場合の有用性について述べた。今後はさらに研究事例を重ねて、時間とコストの視点で従来からの方法と比較し、大規模システムや大量生産システム以外でも新たな検証方法が適用できるかどうか検討することが課題となる。

参考文献

[SY] システム設計検証技術研究会
http://staff.aist.go.jp/k.takahashi/verification.html

[HR]
Michael Huth, Mark Ryan, *Logic in Computer Science: Modelling and reasoning about systems*, Cambridge University Press, 1999

[CGP]
Edmund M. Clarke, Jr., Orna Grumberg, Doron A. Peled, *Model Checking*, MIT Press, 1999

[B+]
B. Berard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L.Petrucci, Ph. Schnoebelen, M. McKenzie, *Systems and Software Verification: Model-Checking Techniques and Tools*, Springer, 2001

[FM] http://archive.comlab.ox.ac.uk/formal-methods.html

[MC] Ken McMillan's HP http://www-cad.eecs.berkeley.edu/~kenmcmil/

[CA] Cadence Design Systems's HP http://www.cadence.com/

[H1] 「自動検針システム　システム仕様書」
本仕様書は、関西電力株式会社の協力のもと、本検証実験のために作成した。

[H2] 「自動検針システム　仕様のモデル化とコード化」

[H3] 「自動検針システム　検査項目と検査の結果」

付録１　：　センタ、GW、WHMの動作仕様（抜粋）

●センタの動作仕様

（1）要求の出力

（a）検針要求の出力

GWに対して任意の時点で検針要求を出力する。

（b）パターン設定要求の出力

GWに対して以下の３つのパターン設定要求を任意の時点で出力する。

●パターン設定要求１
●パターン設定要求２
●パターン設定要求３

（2）略

（3）待機状態

一度要求を出力するとGWから報告を入力するまで待機状態となり、待機中はGWに対して要求は出力しないこととする。

（4）GWからの報告の入力

待機中にGWから報告が出力された場合は、センタとGW間の通信が正常であれば報告を受け付け、待機状態を抜け次の要求をGWに対して出力する。

●GWの動作仕様

（1）要求の受付及び指令データの作成・出力

（a）検針要求の受付

①センタから検針要求があった場合、センタとGW間の通信が正常であれば要求を受け付け、検針指令データを作成しWHMへ出力する。

②センタから検針要求があった場合、センタとGW間の通信が異常であれば要求を受け付けることは出来ず、検針指令データは作成することは出来ないこととととする。

（b）パターン設定要求の受付

①センタからパターン設定要求１があった場合、センタとGW間の通信が正常であれば要求を受け付けパターン設定指令１データを作成しWHMへ出力する。

②センタからパターン設定要求１があった場合、センタとGW間の通信が異常であれば要求を受け付けることは出来ず、パターン設定指令１データは作成することは出来ないこととととする。

上記①、②はパターン設定指令２データ及びパターン設定指令３データについても同様とする。また、GWからWHMへ出力する指令データは、内容が正しい正常データである場合と内容が間違っている異常データである場合があるものとする。

（2）WHMからの返送入力

WHMから返送が出力された場合は、GWとWHM間の通信が正常であれば返送を入力する。（GWとWHM間の通信が異常の場合は（5）の条件に従う）

（3）センタへの報告（正常時）

WHMからの返送を入力した場合は処理が正常に完了したとみなして、その旨（OK）の報告をセンタに出力する。

以下略

●WHMの動作仕様（１／２）

（１）指令データの受付及び返送の作成・出力（ｂ　略）

（ａ）検針指令データの受付

①GWから検針指令データが出力された場合、GWとWHM間の通信が正常であれば指令データを受け付け、返送を作成しGWへ出力する。

②GWから検針指令データが出力された場合、GWとWHM間の通信が異常であれば指令データを受け付けることは出来ず、返送は作成出来ないこととする。

（２）　略

（３）パターンデータの蓄積（　ｂ　略）

（ａ）GWとWHM間の通信が正常の場合（②、④、⑥　略）

GWからパターン設定指令１データ、パターン設定指令２データ、パターン設定指令３データが正常データとして出力された場合、GWとWHM間の通信が正常であれば指令データを受け付け、以下の条件に従ってパターンのデータをメモリ上に蓄積する。

①パターンデータの蓄積は必ずパターン１から開始しなければならない。従ってパターン１のデータを蓄積していない状態で、パターン設定指令２データあるいはパターン設定指令３データを入力した場合は、それらのデータの蓄積は行わない。

③パターン１及びパターン２のデータを蓄積後に、パターン設定指令３データを入力した場合は、パターン３のデータをパターン１及びパターン２のデータと合わせてメモリ上に蓄積し、この時点でメモリ上に全てのパターンデータが揃いメモリ蓄積完了とする。

⑤メモリが「空データ」の状態、あるいは上記②、③、④の後にパターン設定指令１データを入力すると、メモリはパターン１のみのデータで上書きされ、パターン１のデータから蓄積を開始する。

●WHMの動作仕様（２／２）

（４）パターンデータの設定

（ａ）ＷＨＭの電源が正常の場合

全てのパターンデータのメモリ蓄積が完了した場合、WHMの電源が正常の場合は、以下の条件に従って処理を行う。

①全パターンデータのメモリ蓄積が完了した時点で、本体にパターンデータを設定する。

②本体へのパターンデータの設定が完了した時点で、メモリは「空データ」で上書きされる。

（ｂ）ＷＨＭの電源が異常の場合

全パターンデータのメモリ蓄積が完了した場合でも、WHMの電源が異常の場合は本体への設定は出来ないこととする。

付録２　：　項目と状態及びそれぞれに割り当てた変数と値の一覧

表１　項目と状態及びそれぞれに割り当てた変数と値（１／２）

項目	状態	変数	値
GWの電源	正常	GW_P	ON
	異常		OFF
WHMの電源	正常	WHM_P	ON
	異常		OFF
GWとWHM間の通信	正常	Com	1
	異常		0
センタからの要求	要求無し	Req	OFF
	検針要求		M
	パターン設定要求１		P1
	パターン設定要求２		P2
	パターン設定要求３		P3
	待機状態		Wt
GWからWHMへの指令データ	指令無し	Ord_Dt	EMP
	検針指令データ		M
	パターン設定指令１データ		P1
	パターン設定指令２データ		P2
	パターン設定指令３データ		P3
	異常データ		NG
WHMからGWへの返送	処理無し	Ret	EMP
	正常返送		OK
	無応答		NR
センタへの報告	報告無し	Rpt	EMP
	完了		OK
	失敗		NG

表２　項目と状態及びそれぞれに割り当てた変数と値（２／２）

項目	状態	変数	値
パターン蓄積カウンタ	蓄積無し	P_Cnt	EMP
	パターン１のみ蓄積		P100
	パターン１＆２蓄積		P120
	パターン１＆３蓄積		P103
	全パターン蓄積		ALL
パターン設定完了フラグ	完了	P_Set	ON
	未完了		OFF
センタからの要求履歴	履歴無し	Req_Hty	EMP
	パターン１の要求のみ出力済み		P100
	パターン１＆２の要求を出力済み		P120
	パターン１＆３の要求を出力済み		P103
	全パターンの要求を出力済み		ALL
GWからの指令データ履歴	履歴無し	Dt_Hty	EMP
	パターン１の指令データのみ出力済み		P100
	パターン１＆２の指令データを出力済み		P120
	パターン１＆３の指令データを出力済み		P103
	全パターンの指令データを出力済み		ALL