実時間システムの形式仕様記述と試験系列導出

佐藤 文明*　中野 宣政**　水野 忠則*
* 静岡大学情報学部
** 三菱電機株式会社

実時間システムの記述や、マルチメディア通信システムの記述のために、形式記述言語への時間制約表現の拡張が行われている。仕様記述言語LOTOSに対しても、いくつかの時間拡張案が議論されてきた。現在、中心となって議論されている案では、イベントはあくまでも瞬間的で、発生する時間（又は遅延時間）が制約され、イベントは同時には一つしか発生しないという定義である。しかし、現実の世界では相互作用にかかる時間を考える必要がある。上記の案では、相互作用の所要時間を表現するのに、startイベントとendイベントの2つを使って表現することも可能だが、これがLOTOSの並列オペレータで複数組が動作して、他のシステムと同期する場合を考えると、正しい組のstartとendが同期するとは限らない。我々は、このようなstartとendで時間幅を有する相互作用を表現するのではなく、時間幅を持つイベントを定義することが可能な時間拡張を提案する。また、その時間拡張LOTOSからの試験系列の導出について考察する。

1　はじめに

近年、実時間システム、マルチメディアシステムのように、時間制約があるシステムが多く開発されている。時間制約があるシステムを完全に仕様記述するためには、仕様記述言語に時間制約が表現できる必要がある。例えば、仕様記述言語ＳＤＬ[1]には、到着する信号に時間制約を指定する枠組みが組み込まれている。これらの要求に対して、仕様記述言語LOTOS[2]についても時間拡張の標準化作業が進められている。

LOTOSの時間拡張には、いくつかの研究が行なわれ[3][4]、拡張案が議論されてきたが[5][6]、現在はイベントの遅延時間を制約するという方法にまとまりつつある[7]。また、LOTOSの基本的な計算モデルである、イベント順序論理を保持する観点から、イベントは瞬間的なもので、時間幅を有しない方法が提案されている。

しかしながら、現実の世界では、プロセス間のインタラクションやアクションは、ある時間幅を有するものである。従って、イベントに時間幅の制約が記述できる表記方法はユーザにとって便利なものと考えられる。一方で、イベントの時間幅を考慮することは、従来のイベントの順序というものが規定できなくなるとともに、イベントが並列に動作するという複雑な意味定義が必要となる。また、仕様検証や試験系列の導出などができなくなるなどの問題が出る可能性がある。本報告では、LOTOSに対してイベントが時間幅を持つ場合の時間拡張案とその意味定義を検討し、その拡張案に対する試験系列導出に関する考察を行う。

以下、第2章において、現在提案されているLOTOSの時間拡張案について説明する。第3章において、我々が提案するイベントに時間幅を持たせた時間拡張案を論じる。第4章において、我々が提案した時間拡張案からの試験系列方法について検討す

Formal Description and Test Seuquence Generation for Real Time Systems, Fumiaki SATO, Nobumasa NAKANO and Tadanori MIZUNO, Shizuoka University, Mitsubishi Electric Corp.

る。第５章は、本報告のまとめである。

2　既存の時間拡張LOTOS

2.1　時間拡張の基本的な考え方

LOTOSの時間拡張に関する議論が現在ISOで行われている。その基本的な考え方は、LOTOSのイベントの発生するタイミングをある時間幅に制約すること、及びイベントの発生に遅延時間を設定することである。時間はイベントが発生した時に、ある特別な変数に格納することが可能である。

図１に、時間制約の表記方法の一部と対応するイベントの発生のタイミングを示す。

図１のP0では、時間制約なしのイベントの場合であり、発生可能性は、イベントが生起可能になった時刻を０とすると、０から無限大の範囲まで可能である。これをhideオペレータで外部から隠蔽すると(P0')、イベントは同期についての制約がはずされたものとして、即座に内部イベントとして生起すると考える。

P1では、時間制約が付く表記方法であり、イベントａは生起可能になった時間からｔ１時間までの間（生起可能時間帯）に生起可能である。但し、生起可能時間帯を過ぎると、イベントは以後生起できず、stopと解釈される。このイベントを隠蔽すると(P1')、イベントの生起可能時間帯の先頭時刻、すなわち時刻０でイベントは内部イベントとして生起する。

P2では、遅延を含む時間制約の付け方で、イベントａは、生起可能になってから、ｄ時間の間遅延して、ｄからｄ＋ｔ１の時間帯に生起可能である。この時間帯を過ぎするとイベントは発生できず、ｓｔｏｐと解釈される。このイベントを隠蔽すると、イベントの生起可能時間帯の先頭時刻、すなわち時刻ｄで、イベントは内部イベントとして生起する。

P3では、時間制約がついたイベント間の同期をとる場合の解釈について示している。同期をとる二つのイベントの起動時刻、及び時間制約とによって同期がとれる時間帯は変化する。二つのイベントが時間制約を満たし、同期がとれる時間帯が生起可能時間帯となる。これを隠蔽すれば、この生起可能時間帯の先頭時刻が内部イベント発生の時刻となる。

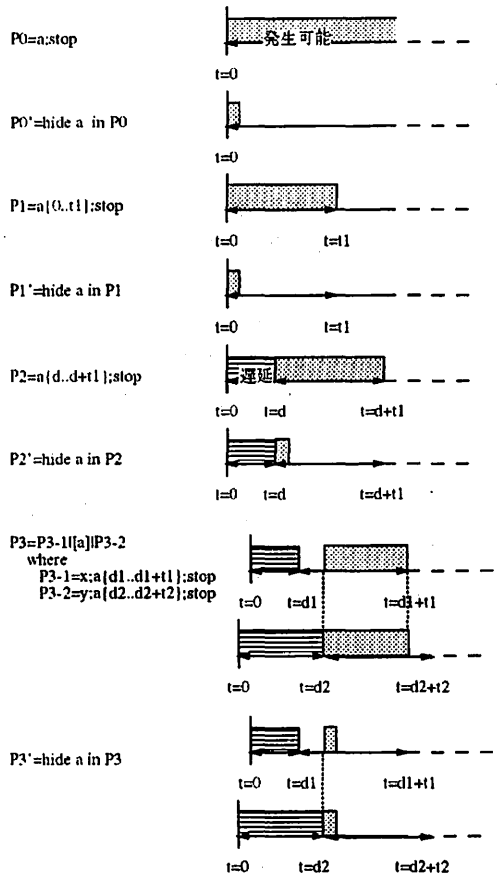


図 1: 時間拡張ＬＯＴＯＳのイベント発生

2.2　時間拡張ＬＯＴＯＳの構文

時間拡張ＬＯＴＯＳの構文の概要を表１に示す。但し、説明の都合上、データ部分と一般選択や遅延演算などを省略してある。

名前	構文
Stop	stop
Observable Action Prefix	g{t in T};B
Internal Action Prefix	i{t in T};B
Exit	exit(T)
Choice	B1 [] B2
Interleave Composition	B1 |[G]| B2
Hiding	hide G in B
Enabling	B1 >> B2
Disabling	B1 [> B2

表 1: 時間拡張ＬＯＴＯＳの構文

イベントの時間制約は、$g\{t\ in\ T\}$ という表記をとる。ここでＴはイベントｇが発生可能な時間幅を示しており、ｔには発生した時間が格納される。ｔやＴは、ある特別なドメイン（時間ドメイン）の変数であり、イベントが発生する際に書き込まれ、あとは読み出しのみ使用可能な変数と定義している。“ｔ in ”あるいは“in Ｔ”は省略可能であり、省略すればそれぞれ発生時刻の格納がないもの、また制約がないものを意味する。

2.3 時間拡張ＬＯＴＯＳの意味定義

時間拡張ＬＯＴＯＳの意味定義の概要を表２に示す。hide オペレータの意味定義で、隠蔽されたイベントは、同期に関する制約がないこととし、なるべく早期に内部イベントとして発生するものとする意味が定義されている。

2.4 時間拡張ＬＯＴＯＳの問題点

現実のシステムは、相互作用においては、必ずいくらかの時間を消費しており、それらを厳密に記述する必要がある。すなわち、イベントの時間幅という考え方が必要になる。しかし、現在のＬＯＴＯＳの枠組みで

stop -d-> stop (0<d)
g{t in 0 .. d};P -g-> P g{t in d+d1 .. d+d2};P -d-> g{t in d1 .. d2};P (d1<d2) g{t in d1 .. d2};P -d-> stop (d1<=d2<d) g{t in 0 .. d+d1};P -d-> g{t in 0 .. d1};P
i{t in 0 .. d};P -i-> P i{t in d+d1 .. d+d2};P -d-> i{t in d1 .. d2};P (d1<d2) i{t in 0 .. d+d1};P -d-> i{t in 0 .. d1};P
exit{0 .. d} -δ->stop exit{d+d1 .. d+d2} -d-> exit{d1 .. d2} exit{d1 .. d2} -d-> stop (d1<=d2<d) exit{0 .. d+d1} -d-> exit{0 .. d1}
P -a-> P' / P [] Q -a->P' ; Q -a-> Q' / P[]Q -a->Q' ; P-d-P', Q-d->Q' / P[]Q-d->P'[]Q'
P-a->P' / P|[G]|Q -a-> P'|[G]|Q (a is not included in GU{δ}) ; Q-a->Q' / P|[G]|Q -a-> P|[G]|Q' (a is not included in GU{δ})
P-a->P', Q-a->Q' / P|[G]|Q -a-> P'|[G]|Q' (a is included in GU{δ}) ; P-d->P', Q-d->Q' / P|[G]|Q -d-> P'|[G]|Q'
P-a->P' / hide G in P -a-> hide G in P' (a is not included in G) ; P-a->P' / hide G in P -i-> hide G in P' (a is included in G)
P-d->P', (Gに含まれるｇが、任意のｄ' ＜ｄにおいて、発生しない) / hide G in P -d-> hide G in P'
P-a->P' / P >> Q -a-> P' >> Q (a != δ) ; P-a->P' / P >> Q -a-> Q (a =δ)
P-d->P', (任意のｄ' ＜ｄにおいて、δが発生しない) / P >> Q -d-> P' >> Q
P -a-> P' / P [> Q -a->P'[>Q (a != δ) ; Q -a-> Q' / P[>Q-a->Q' ; P-a->P' / P[>Q-a->P' (a =δ) ; P-d-P', Q-d->Q' / P[>Q-d->P'[>Q'

表 2: 時間拡張ＬＯＴＯＳの意味定義

は、イベントは時間を持たないもので、それによってイベント順序論理を構成している。従って、イベント順序論理による意味定義を保存したままでは、イベントの時間幅の記述は難しい。

現在のＬＯＴＯＳのイベントの枠組みを使ってイベントの時間幅を記述方式として、時間幅のあるイベントの開始点と終了点を時間幅のないイベントで記述する方法がある。例えば、g-start{t in T}; g-end{t' in T'}; stop と記述する。しかし、このように本来は一つのイベントが分割された場合、並列合成した場合のイベントのインタリーブにより、必ずしも正しい分割イベントが順番通り起動するとは限らない。

また、時間拡張ＬＯＴＯＳの意味定義では、通常のイベントはある瞬間に一つしか起きていないが、時間イベントについては完全に並行に発生するという複雑なモデルになっている。現実には、通常のイベントについても並行に発生することがあり、すべてが並行に発生する定義を行えば、統一した意味定義が可能と考えられる。

3　本研究での拡張

3.1　時間幅を持つイベント

上記の問題点を解決する方法として、時間幅を持つイベントを記述する枠組みを提案する。特徴は、以下のようになる。

（1）　イベントは、発生時刻の制約とともに、発生時間幅の制約が付く。

（2）　並行処理は、インタリーブ動作と、完全な並行動作との２種類となる。

（3）　時間イベントと通常イベントという区分はなく、時間＋生起イベント集合から動作仕様が定義される。

3.2　時間幅拡張ＬＯＴＯＳの構文

時間幅を拡張したＬＯＴＯＳの構文を表３に示す。但し、表１と同様に、説明の都合上、データ部分と一般選択や遅延演算などを省略してある。

名前	構文
Stop	stop
*Observable Action Prefix	g{t in T}<u in U;B
*Internal Action Prefix	i{t in T}<u in U>;B
Exit	exit{T}
Choice	B1 [] B2
Interleave Composition	B1 |[G]| B2
*Parallel Composition	B1 |<G>| B2
Hiding	hide G in B
Enabling	B1 >> B2
Disabling	B1 [> B2

表 3: 時間幅拡張ＬＯＴＯＳの構文

時間幅を拡張したイベントの時間制約は、$g\{t\ in\ T\} < u\ in\ U >$ という表記をとる。ここでＴはイベントｇが発生可能な時間幅を示しており、ｔには発生した時間が格納される。同様に、Ｕはイベントｇが発生している時間幅の制約を与え、ｕにその時間幅が格納される。ｔ,u, Ｔ及びＵは、ある特別なドメイン（時間ドメイン）の変数であり、イベントが発生する際に書き込まれ、あとは読み出しのみ使用可能な変数と定義している。“ｔ in ”、“u in” あるいは “in T”、“in U” は省略可能であり、省略すればそれぞれ発生時刻の格納がないもの、また制約がないものを意味する。

インタリーブ合成の “$B1|[G]|B2$” 演算は、B1 に含まれるイベントと B2 に含まれるイベントが、ゲート集合Ｇに属していれば同期をとり、それ以外の場合はインタリーブ型の並行動作を行うことを示している。パラレル合成の “ $B1| < G > |B2$” 演算は、B1 に含まれるイベントと、B2 に含まれる

イベントがGに含まれていない限り完全に並行に動作することを示している。

また、イベントの同期にあたっては、同期するイベントの時間幅は同じになる必要がある。また、時間幅の最低値が指定されている場合、最低値を満たせないイベントとは同期できない。

3.3 時間幅拡張ＬＯＴＯＳの意味

時間幅を拡張したＬＯＴＯＳの意味定義を表4に示す。この定義の特徴は、LTS（ラベル付遷移システム）のラベルに、従来はイベント（通常イベント、あるいは時間イベント）のみが付与されていたのに対して、イベント集合と時間幅の組が付与されることである。イベント集合が空集合の場合は、従来の時間イベントと等価であり、時間幅0の場合は時間幅を持たない従来の通常イベントと同じに扱える。

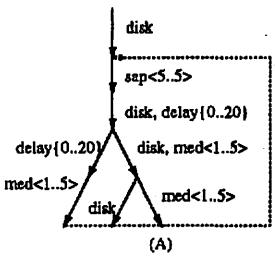
3.4 イベント集合木

従来の時間拡張ＬＯＴＯＳでは、通常イベントは同時に起こることはないため、イベント木によって表現できる。イベント木の枝には、イベント名と時間制約や選択条件などの制約が付与されることになる。

時間幅拡張のＬＯＴＯＳは、従来のＬＯＴＯＳのイベント木によって記述するのは困難である。それは、イベントに時間幅を定義することでイベント同士の時間的な重なりを認めているためである。このようなイベントによる木を表現するため我々は、イベント集合木を提案する。

イベント集合木では、各枝がイベントの集合を表している。但し、各枝が一つのイベント（集合）の開始と終了を意味しない。この枝は、イベントの実行状態（実行されているイベントの集合の要素）の変更を意味している。しかし、従来のイベント木についても、同じ枠組で記述することが可能である。即ち、実行されているイベント集合の要素が常に一つであるイベント集合木と考えることができる。従って、イベント集合木は、従来のイベント木の上位互換となっている。

```
process MEDIUM[disk, sap, med]:=

  stream-sender[disk, sap]
  |<sap>|
  medium[sap, med]

where

  process stream-sender[disk, sap]:=
    disk{0..0}; sap{0..0}<5..5>; stream-sender[disk, sap]
  endproc

  process medium[sap, med]
    sap{0..0}<5..5>; med{0..20}<1..5>; medium[sap, med]
  endproc

endproc
```



(A)

図 2: 時間幅拡張ＬＯＴＯＳのイベント集合木

4 試験と等価性について

4.1 イベント木からの試験系列導出

ここで述べる試験とは、時間幅記述が拡張されたＬＯＴＯＳ仕様に基づいて作成されたシステムを試験するものである。試験対象システムは、時間幅拡張ＬＯＴＯＳの相互作用インタフェースを持つと仮定する。従って、ここで述べる試験は、試験システムと試験対象システムを同期オペレータにより完全同期で動作させることで実現できる。

基本的に、試験系列は試験対象システムの仕様からイベント木をたどることによって生成できる。イベント木には、そのイベントが発生するための制約条件が記述され

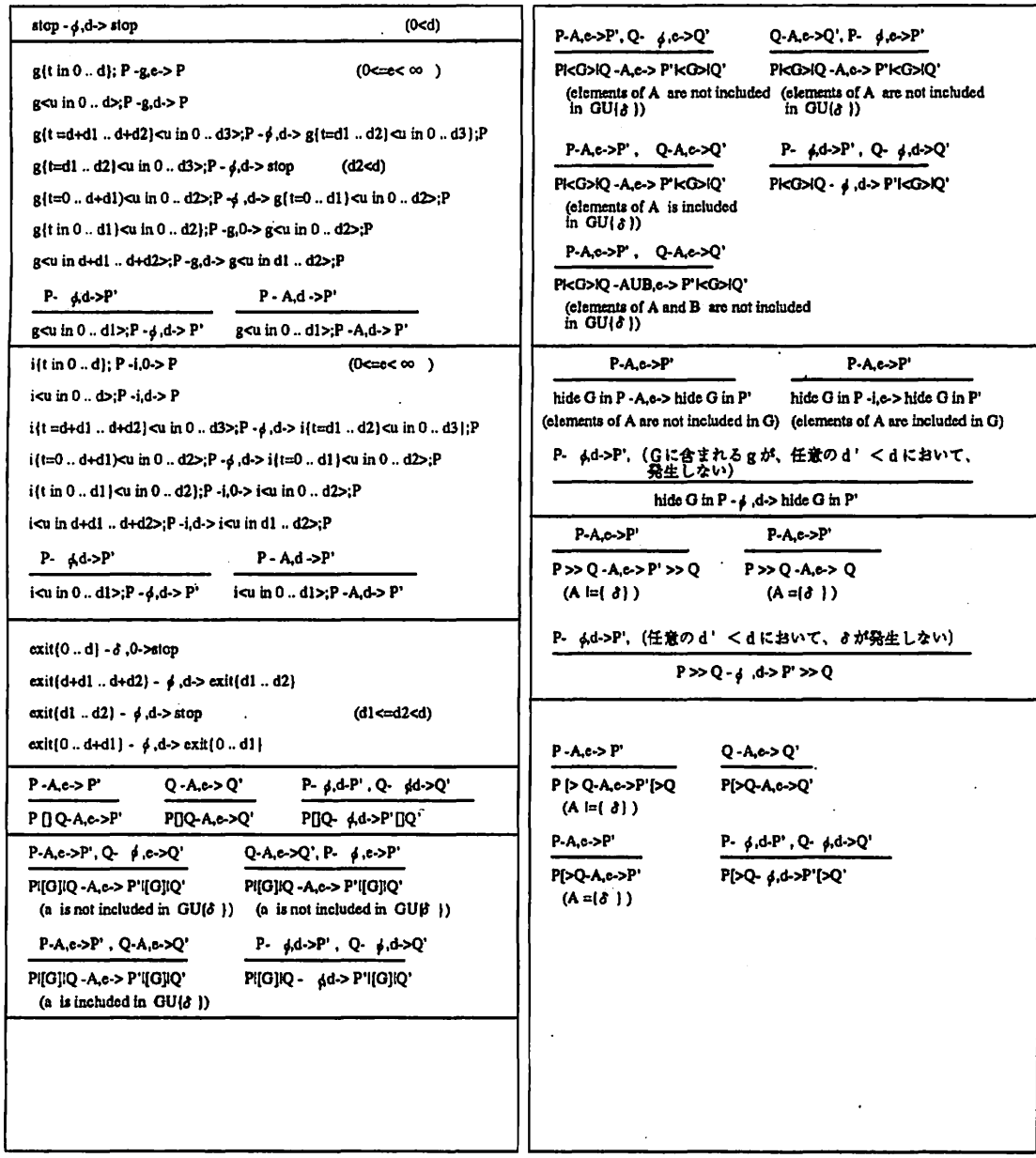


表 4: 時間幅拡張ＬＯＴＯＳの意味定義

る。よって、この条件をすべて満たすように、条件を設定すれば良い。ここで、文献[8]では、条件の記述を整数1次不等式に限定することで、線形計画法を利用することを提案している。

同様な手法をイベント集合木に適用することで、試験系列を導出することができる。図2において、(A)で示したノードまでの試験系列は、ルートノードから(A)までの経路にあたるイベント集合木の枝をたどることによって、求まる。即ち、

```
disk
sap<5..5>
disk, delay{0..20}
disk, med<1..5>
med<1..5>
```

が求まる。ここで、条件として、sapに関する時間幅、medに関する起動時間、及び時間幅が条件として設定されている。diskに関しては時間制約は設定されていないが、2度目のdiskについては、medと同時でかつ時間的に完全に含まれるように起こる条件が、系列から導かれる。これらの条件を不等式で表現して、線形計画法を適用すれば、時間制約を満足する条件を求めることができ、試験系列が導出できる。上記の例では、結局試験系列としては、次のようなものが求められることになる。

```
disk<1..1>
sap<5..5>
disk<13..13>, delay{10..10}
disk<13..13>, med<5..5>
med<5..5>
```

これらの条件について、線形計画法で求まらない場合、その時間制約に矛盾があることになる。従って、そのようなノードには推移できないと判断できる。

4.2 バイシミュレーション等価性

LOTOSにおける等価性は、バイシミュレーション等価性として定義されている。時間拡張LOTOSにおいても、その等価性が定義された。

二つのシステムにおける状態B1,B2と、任意のイベントa（通常イベント及び時間イベント）を考えたとき、

(1) もし $B1 - a- > B1'$ であれば、あるB2'が存在し、$B2 - a- > B2'$ となる。

(2) もし $B2 - a- > B2'$ であれば、あるB1'が存在し、$B1 - a- > B1'$ となる。

このような関係が成り立つ状態の対応づけが存在するとき、強バイシミュレーション等価と定義している。

また、$= a =>$ を任意の内部イベントを含む通常イベントあるいは時間イベントとするとき、

(1) もし $B1 = a => B1'$ であれば、あるB2'が存在し、$B2 = a => B2'$ となる。

(2) もし $B2 = a => B2'$ であれば、あるB1'が存在し、$B1 = a => B1'$ となる。

このような関係が成り立つ状態の対応づけが存在するとき、弱バイシミュレーション等価と定義している。

これに対して、我々は時間幅と並列動作が拡張され、意味定義においても時間とイベント集合の組み合わせによって定義している。従って、次のような定義をバイシミュレーション等価性として定義する。

二つのシステムにおける状態B1,B2と、任意のイベント(A,e)（イベント集合＋時間幅）

(1) もし $B1 - A, e- > B1'$ であれば、ある B2' が存在し、$B2 - A, e- > B2'$ となる。

(2) もし $B2 - A, e- > B2'$ であれば、ある B1' が存在し、$B1 - A, e- > B1'$ となる。

このような関係が成り立つ状態の対応づけが存在するとき、強バイシミュレーション等価と定義する。

また、$= A, e =>$ を任意の内部イベントを含む通常イベントあるいは時間イベントとするとき、

(1) もし $B1 = A, e => B1'$ であれば、ある B2' が存在し、$B2 = A, e => B2'$ となる。

(2) もし $B2 = A, e => B2'$ であれば、ある B1' が存在し、$B1 = A, e => B1'$ となる。

このような関係が成り立つ状態の対応づけが存在するとき、弱バイシミュレーション等価と定義する。

5 まとめ

時間制約を LOTOS に拡張する方法として、従来のイベントの起動時間に制約をつける方法に加え、イベントの時間幅という制約を記述できるように拡張する方法を提案した。これにより、現実の世界で起こっているインタラクションにかかる時間の制約を厳密に規定することができる。

また、時間幅の拡張により、従来のイベント木による表記が困難であることから、新たなイベント集合木を提案し、それに基づいて試験系列が導出できることを示した。また、等価性についても、新たな意味定義方法に沿ったバイシミュレーション等価性の定義を与えた。

今後は、等価性に関する定義の妥当性について検証する。また、より効率的な試験系列の導出方法についても検討する。

参考文献

[1] ITU-T: SDL, Specification and Description Language, ITU-T Z.100.

[2] ISO Standard 8807: LOTOS, a Formal Description Technique based on Temporal Ordering of Obsevational Behavior, 1988.

[3] J.P.Courtiat and R.C. Oliveira: RT-LOTOS and its application to multimedia protocol specification and validation, Int. Conf. on Multimedia and Networking, 1995.

[4] A.Nakata, T.Higashino, K.Taniguchi: LOTOS enhancement to specify time constraint among no-adjacent actions using 1st-order logic, FORTE'93, 1993.

[5] C.Miguel, A.Fernandez, and L.Videller: Extending LOTOS towards performance evaluation, FORTE'93,1993.

[6] L.Leonard and G.Leduc: An Enhanced Version of Timed LOTOS and its Application to a Case Study, FORTE'93, 1993.

[7] ISO/IEC JTC1/SC21/WG7 N1053 : Revised Working Draft on Enhancements to LOTOS(V3), ISO, 1996.

[8] 李、東野、谷口：データを含む LOTOS 記述に対するテスト系列の自動生成の一手法、電子情報通信学会論文誌 B-I, Vol.J75-B-I, No.11, 1992.