Formal Verification of Temporal Properties for Reduced Overhead in Grid Scientific Workflows*

Junwei Cao(曹军威) [1,2], *Senior Member*, *IEEE*, Fan Zhang(张帆) [3], *Student Member*, *IEEE*, Ke Xu(许可)[4], Lianchen Liu(刘连臣)[3] and Cheng Wu(吴澄)[3]

[1]Research Institute of Information Technology, Tsinghua University, Beijing 100084, P. R. China

[2]Tsinghua National Laboratory for Information Science and Technology, Beijing 100084, P. R. China

[3]National CIMS Engineering and Research Center, Tsinghua University, Beijing 100084, P. R. China

[4]Research engineer, Morgan Stanley, Shanghai, 200002, P. R. China

E-mail: jcao@tsinghua.edu.cn; zhang-fan07@mails.tsinghua.edu.cn; Ke.Sh.Xu@morganstanley.com; liulianchen@tsinghua.edu.cn; wuc@tsinghua.edu.cn

Abstract With quick development of grid techniques and growing complexity of grid applications, it is becoming critical for reasoning temporal properties of grid workflows to probe potential pitfalls and errors, in order to ensure reliability and trustworthiness at the initial design phase. A State Pi Calculus is proposed and implemented in this work, which not only enables flexible abstraction and management of historical grid system events, but also facilitates modeling and temporal verification of grid workflows. Furthermore, a Relaxed Region Analysis (RRA) approach is proposed to decompose large scale grid workflows into sequentially composed regions with relaxation of parallel workflow branches, and corresponding verification strategies are also decomposed following modular verification principles. Performance evaluation results show that the RRA approach can dramatically reduce CPU time and memory usage of formal verification

Keywords Grid Computing; Workflow Management; Formal Verification; State Pi Calculus

1 Introduction

Grid Computing is becoming a mainstream technology for cross-domain management and sharing of computational resources [1]. Grid workflows [2], a composition of various grid services according to prospective processes, have become an important paradigm for the problem solving in various scientific and industrial domains, e.g. gravitational wave data analysis [3], biomedical simulation [4], and banking [5].

The quick growing complexity of grid applications and systems calls for the implementation of reliable and trustworthy grid workflows according to specific scientific criteria or business regulations, which has become an urgent research issue. In addition to existing grid enabling techniques, e.g. job scheduling, workflow enactment and resource location, various grid

* PAPER CLASSIFICATION

This work is supported by National Science Foundation of China (grant No. 60803017) and Ministry of Science and Technology of China under National 973 Basic Research Program (grants No. 2011CB302505 and No. 2011CB302805) and National 863 High-tech R&D Program (grants No. 2008AA01Z118 and No. 2008BAH32B03). Fan Zhang is supported by IBM 2010-2011 PhD Fellowship.

ensuring techniques [6], e.g. temporal reasoning [7], are gaining more and more attention from grid community. These techniques are devoted to guarantee large scale grid workflows follow exactly requirements of domain users.

Delivering reliable web service applications using petri-net or related techniques are fully investigated [8][9][10]. Quality of Services under web service configuration are discussed [11][12]. All these ensuring techniques are used to improve compatibility and reliability of grid workflow applications. This work differentiate itself from those works as using Pi Calculus based methods.

It has already been widely recognized that techniques, like formal verification based temporal reasoning [13][14][15], are becoming more and more important for Web Services based systems in probing their potential errors and enhancing their reliability. How process algebras can be applied to model and reason the choreography of web services is discussed in [16]. Regarding grid system formalization, the Abstract State Machine based formalism is applied in [17] to distinguish grid features from traditional distributed systems. Other areas that this method could be used include software reuse and compatibility checking [18] and scale-free web service composition or decomposition applications[19].

Pi calculus was first proposed by R. Milner for its intrinsic combinability and mobility together with its natural description for open communication system [20]. This method is now accepted for its sound theoretical system and widely used in formal modeling, verification and validation. State Pi calculus, an extension of Pi calculus, is implemented in this work to further strengthen its capability to manage the life-cycle of system states. The proposed calculus not only enables the flexible abstraction and management of historical system events, but also facilitates the modeling and verification of grid service based workflows.

While there are previous attempts in the study of grid verification techniques, the performance for verification itself is still a bottleneck for probing all potential pitfalls and errors in grid workflows. Especially for large scale and dynamically evolving scientific workflows, implementation of such formal verification processes have to be of low overhead in terms of verification time and memory demand to be applicable in real world grid environments. Performance improvement is also focused in this work. A Relaxed Region Analysis (RRA) approach is proposed to divide-and-conquer global verification of a very large scale scientific workflow in LIGO applications into local verifications on its sub grid workflow models.

Decomposition is a common technique used for handling complex systems in order to exponentially decrease system dimensions for overhead reduction. While application-specific decomposition strategies have been investigated in [21] for carrying out computational tasks in grid environments, our work is addressing a more general decomposition

approach for grid workflows by studying their local and global process structures. Also we focus on how the correctness of a grid workflow, instead of a grid infrastructure itself, can be efficiently fulfilled. The RRA approach further studies how the formal verification is decomposed with the decomposition of grid workflow to form a complete region analysis method. Besides, it also allows the relaxation of parallel branches in grid workflows to achieve better decomposition results and verification performance.

The approach is implemented using a Pi Calculus based formal modeling and verification environment for grid workflows with NuSMV2 [22] as its engine. Three concrete application scenarios from gravitational wave data analysis [23] are provided, which are currently the most classic grid-enabled scientific applications in the United States. While the complexity of a grid workflow grows exponentially when the number of its involving services and their interdependencies increases, the RRA approach can dramatically reduce overhead such as CPU time and memory usage of formal verification processes, as illustrated by quantitative performance results included in this work.

The rest of the paper is organized as follows. In Section II, the state Pi calculus and corresponding formalism of different structures are introduced. Section III provides detailed information on grid workflow modeling and the concept of standard regions. Corresponding verification decomposition is described in Section IV. The RRA approach is proposed in Section V and the implementation of RRA with performance evaluation results are investigated in Section VI. Section VII concludes the paper.

2 State Pi Calculus and Formalism

2.1 State Pi Calculus

State Pi calculus, as an extension to the original formalism framework of Pi Calculus, has three main features:

1) Utilize historical information to restrain and analyze process evolution.

2) Provide flexible abstraction of activities and communications in processes via administration of status proposition.

3) March general principles in WSDL to extend web services.

It is defined as follows:

$$
\begin{aligned}
P &::= \sum_{i=1}^{n} \alpha_i\{StateExp\}.P_i \mid (new\ x)P \mid !P \mid P \mid Q \mid \phi P \mid A(y_1,...,y_n) \mid 0 \\
\alpha_i\{StateExp\} &::= \overline{x} < \widetilde{y} > \{StateExp\} \mid x(\widetilde{y})\{StateExp\} \mid \tau\{StateExp\} \\
\phi &::= [x = y] \mid eval(Prop, valueset) \mid \phi \wedge \phi \\
StateExp &::= (StateOp, S) \mid StateExp, StateExp \\
StateOp &::= + \mid - \mid ++ \mid -- \\
S &::= (iden, trueset) \\
iden &::= x \mid iden.iden
\end{aligned}
$$

The fundamental concept of State Pi Calculus is the names, which are used to express atomic interactive actions in a system. A system in State Pi Calculus evolves through the operators, including *composition* ‘|’, *choice* ‘+’, *guard* ‘.’, *match* ‘[]’, *restriction* ‘new’ and *replication* ‘!’. Every operator implements one kind of the relation $\Re$: *SysState* $\times$

StateOp $\times S \rightarrow$ *SysState*, which is strictly defined in *StateExp*. The current system state *SysState* together with the state object (*StateOp* and *S*) evolve into a new system state *SysState*. All the states related implementations are new in this new method.

1) An output action ($\bar{x}\langle\tilde{y}\rangle\{StateExp\}.P$): This means outputting $\tilde{y}$ through $\bar{x}$ with system behaviors evolved into *P* and system states evolved into a new state based on the predefined expression in *StateExp*. For example, in a communication system $\bar{x}$ can be viewed as an output port and $\tilde{y}$ the output data, *StateExp* is the corresponding transition of the states.

2) An input action ($x(\tilde{y})\{StateExp\}.P$): This means inputting $\tilde{y}$ through x with system behaviors evolved into *P* and system states evolved into a new state.

3) A silent action ($\tau\{StateExp\}.P$): The system behavior evolves into *P* with internal actions instead of interactions with the environment and system states evolved into a new state.

4) A composition (*P*|*Q*): Processes *P* and *Q* are independent, or synchronize with each other via an identical port.

5) Choice (*P*+*Q*): Unpredictable execution of *P* or *Q*.

6) March ([*x*=*y*]*P*): If *x* matches *y*, the system behavior evolves into *P*. Otherwise no actions happen.

7) Restriction ((*new x*) *P*): *x* is a new name within the process *P*.

Replication (!*P*): An infinite composition of process *P*.

2.2 Formalism of Activities and Control Structures

In this section, BPEL4WS (Business Process Execution Language for Web Services) is considered as an example to describe how State Pi calculus can be used for formal representation of workflow activities and control structures.

Four basic activities from BPEL4WS, *Receive*, *Send*, *Invoke* and *Assign*, can be defined as follows using State Pi calculus.

$$Invoke(start, get, port_1, port_2, done) =_{def} \; start.get(v:t).\overline{port_1} < v:t > .port_2(s).\overline{done}$$

$$Receive(start, port_2, set, done) =_{def}$$
$$start\{++,\{msgPort,\{port_2\}\}\}.port_2(v:t)\{--,\{msgPort,\{port_2\}\}\}.\overline{set} < v:t > .\overline{done}$$

$$Reply(start, get, port_1, done) =_{def} \; start.get(v:t).\overline{port_1} < v:t > .\overline{done}$$

$$Assign(start, get_1, set_2, done) =_{def} \; start.get_1(v:t).\overline{set_2} < v:t > .\overline{done}$$

$$Empty(start, done) =_{def} \; start.\overline{done}$$

$$Var_V =_{def} \; Var_{V0}(set, get)$$

$$Var_{V0}(set, get) =_{def} \; set(x_1:t)\{++,\{V.bSize,\{x_1\}\}\}.Var_{V1}(set, get, x_1)$$

$$Var_{V1}(set, get, x_1) =_{def} \; set(x_2:t_2)\{++,\{V.bSize,\{x_2\}\}\}.Var_{V2}(set, get, x_1, x_2) +$$
$$\overline{get} < x_1:t_1 > \{--,\{V.bSize,\{x_1\}\}\}.Var_{V0}(set, get)$$

.....

$$Var_{Vn}(set, get, x_1, ..., x_n) =_{def} \; \overline{get} < x_n:t_n > \{--,\{V.bSize,\{x_n\}\}\}.Var_{Vn-1}(set, get, x_1, ..., x_{n-1})$$

The formalism of six control structures in BPEL4WS, *Sequence*, *While*, *Flow*, *Switch*, *Pick* and *Link*, is described as follows.

1) Formalism of the *Sequence* structure

The *Sequence* structure defines sequential relations

among executions in a grid workflow:

$$Sequence(fn_d(Act_1), fn_s(Act_2))$$
$$=_{def} \ Act_1 ; Act_2 =_{def} \ (new\ start_{Act_2})(\{start_{Act_2} / done_{Act_1}\}Act_1 \mid Act_2)$$

2) Formalism of the *While* structure

The *While* structure defines repeat invocation of one or a group of services in a grid workflow under certain conditions:

$$While(fn_{sd}(Act), start_{while}, done_{while}) =_{def} \ new\ start_{Act}\ done_{Act}(start_{while}.$$
$$([eval(C,\{t\})](\overline{start_{Act}} \mid Act \mid done_{Act}.(\overline{start_{while}} \mid While)) + [eval(C,\{f\})]\overline{done_{while}}))$$

3) Formalism of the *Flow* structure

The *Flow* structure defines synchronization of parallel execution, completion among service activities and structures in a grid workflow:

$$Flow(fn_{sd}(Act_1), ..., fn_{sd}(Act_m), start_{Flow}, done_{Flow}) =_{def}$$
$$(new\ start_{Act_1} \ ... \ start_{Act_m}\ done_{Act_1} \ ... \ done_{Act_m}\ ack\ ack')$$
$$(start_{Flow}.Starter \mid Act_1 \mid ..... \mid Act_m \mid Acker \mid \overline{ack'.done_{Flow}})$$
$$Starter(start_{Act_1}, ..., start_{Act_m}) =_{def} \ \overline{start_{Act_1}} \mid ..... \mid \overline{start_{Act_m}}$$
$$Acker(done_{Act_1}, ..., done_{Act_m}.ack, ack') =_{def} \ done_{Act_1}.\overline{ack} \mid ..... \mid done_{Act_m}.\overline{ack} \mid \underbrace{ack.....ack}_{m}.\overline{ack'}$$

4) Formalism of the *Switch* structure

The *Switch* structure defines one conditional choice in a grid workflow:

$$Switch(fn_{sd}(Act_1), fn_{sd}(Act_2), start_{Switch}, done_{Switch}) =$$
$$_{def}(new\ start_{Act_1}\ start_{Act_2}\ done_{Act_1}\ done_{Act_2})$$
$$(start_{Switch}.([eval(C_1,\{t\})]\overline{start_{Act_1}}\{(++,\{Branch,\{Act_1\}\})\} \mid Act_1 \mid done_{Act_1}.\overline{done_{Switch}} +$$
$$[eval(C_1,\{f\}) \wedge eval(C_2,\{t\})]\overline{start_{Act_2}}\{(++,\{Branch,\{Act_2\}\})\} \mid Act_2 \mid done_{Act_2}.\overline{done_{Switch}}))$$

5) Formalism of the *Pick* structure

The *Pick* structure defines execution selection among different services and structures in a grid workflow based on message triggers:

$$Pick(fn_{sd}(Act_1), fn_{sd}(Act_2), port_{p1}, port_{p2}, timeout, start_{Pick}, done_{Pick}) =_{def}$$
$$(new\ start_{Act_1}\ start_{Act_2}\ done_{Act_1}\ done_{Act_2})(start_{Pick}.$$
$$(port_{p1}\{++,\{msgPort_1,\{port_{p1}\}\}\}.$$
$$start_{Act_1}\{(++,\{Event,\{Act_1\}\}),(--,\{msgPort_1,\{port_{p1}\}\})\} \mid Act_1 \mid done_{Act_1}.\overline{done_{Pick}} +$$
$$port_{p2}\{++,\{msgPort_2,\{port_{p2}\}\}\}.$$
$$start_{Act_2}\{(++,\{Event,\{Act_2\}\}),(--,\{msgPort_2,\{port_{p2}\}\})\} \mid Act_2 \mid done_{Act_2}.\overline{done_{Pick}}) +$$
$$timeout\{(++,\{Event,\{Timeout\}\})\}.\overline{done_{Pick}})$$

6) Formalism of the *Link* structure

The *Link* structure imposes synchronization constraints on activities in a grid workflow. Each *Link* has a source and target activity, which restricts that the target activity can only be executed after the source activity is done.

$$Link_i(done_{in}, neg_{in}, ack, nack) =_{def} \ EvalTransCondition_i(done_{in}, neg_{in}, ack, nack)$$
$$EvalTransCondition_i(done_{in}, neg_{in}, ack, nack) =_{def} \ done_{in}.\overline{ack} + neg_{in}.\overline{nack} \qquad i = 1, ..., n$$
$$Links(done_{in}, neg_{in}, done_{links}, deathpath) =_{def} \ (new\ ack\ nack)$$
$$(Link_1 \mid ... \mid Link_n \mid \underbrace{ack.(... .(ack}_{n}.\overline{done_{links}} + nack.\overline{deathpath})...) + nack.\overline{deathpath}))$$

$$ActWithLinks(freeN) =_{def} \ start.(done_{links}.new\ t\ f(\overline{evaljoin} < t, f >.($$
$$t.(new\ start_{Act}\ done_{Act})(\overline{start_{Act}} \mid Act \mid done_{Act}.\overline{done}.\prod\overline{done_{out}}) +$$
$$f\{++,\{Exception,\{Act\}\}\}.\overline{throw} < joinfailure >)) + deathpath.\prod\overline{neg_{out}})$$
$$freeN = \{start, done, done_{links}, done_{out}, neg_{out}, deathpath,$$
$$evaljoin, fault, joinfailure, fn_{sd}(Act)\}$$

3 Grid Workflows and Standard Regions

3.1 Preliminary Constraints

Considering that there are various grid workflow specification languages, common notations used in this paper are provided in Fig. 1 to visually represent a grid workflow model. Modeling elements in Fig. 1 cover as many existing workflow languages (e.g. BPEL4WS) as possible.

To prevent possible construction of unstructured grid workflows, syntactical constraints are defined as a unified basis for our region analysis, which is concluded from soundness criteria (no deadlocks and no multiple service activity instances on the same service activity) [24].

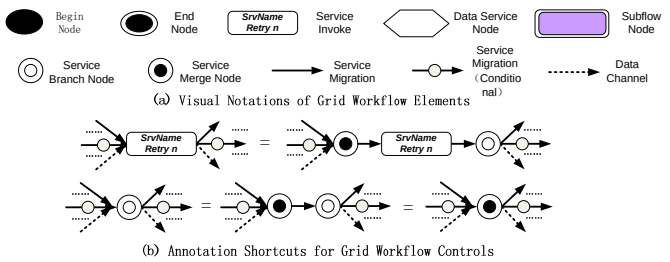


Fig. 1. Visualization of grid workflow elements

Constraint 1: We refer a Srv&Ctrl node to a Grid Service Activity, Subflow or Control node and refer a SrvFlow node to a Srv&Ctrl or Data Service node.

Constraint 2: Each grid workflow has exactly one explicit Begin node and End node (which will be later relaxed in our RRA approach).

Constraint 3: Every Srv&Ctrl node must be syntactically reachable from the Begin node and can reach the End node by transitions (i.e., no dangling Grid Service Activity, Subflow, or Control nodes).

Constraint 4: Each transition has exactly one source / target Srv&Ctrl node. Each data channel has at most one source / target SrvFlow node (with one of them must be a Data Service Node). Therefore, a Data Channel is called fluent if and only if (IFF) its source / target Data Service node is reachable from the Begin node / can reach the End node.

Constraint 5: Multiple inputs and outputs are allowed for a Grid Service Activity and Control node. Their equivalent semantics are illustrated in Fig. 1(b).

Constraint 6: Arbitrary cycles are allowed as long as no unstructured workflow models are caused.

Fig. 2 illustrates an example gravitational wave data analysis workflow SF1 based on visual notations provided in Fig. 1.

3.2 Standard Regions

The most common temporal relations in formal verification is the combinations of "what will eventually / always happen in the future?" and "something will hold until an event is received". Therefore the idea is to encapsulate a grid workflow into separate sub workflows such that important relations can be directly implied between them. Consequently, instead of global reasoning of the whole workflow, which is quite costly, it can be equivalent to locally investigate behaviors of sub workflows.

Denote $N_1 \rightarrow N_2 \ldots \rightarrow N_m$ to be a directed path from node N1 to Nm in a grid workflow. Note that by $N_1 \rightarrow N_2$, it only means that N_1 and N_2 are syntactically connected by transitions or data channels and N_2 / N_1 is the target / source of the transition or data channel. Consequently the structural information of a workflow is our primary concern in the definition of regions.

Definition 1 (Region): Two Srv&Ctrl nodes or Begin / End nodes Nhead and Ntail form a region in a grid workflow Γ, denoted by $\{N_{head}, N_{tail}\}$, IFF: (1) $\nexists$ $N_{head} \rightarrow N_1 \rightarrow \ldots N_m \rightarrow$ End where End is the End

node of Γ, and N_i != N_{tail} ($i=1,\ldots,m$); (2) $\nexists$ Begin $\rightarrow N_1 \rightarrow \ldots N_m \rightarrow$ Ntail where Begin is the Begin node of Γ, and N_i != N_{head} ($i=1,\ldots,m$).

More intuitively, a region $\{N_{head}, N_{tail}\}$ specifies a structure in which node N_{head} will always reach N_{tail} in order for it to reach the *End* node in Γ (and vice versa). For example, in Fig. 2 {*TrigBank_H2_3, thIncaII_L1H2*} is a region while {*sInca_L1H1, thIncaII_L1H2*} is not. The whole grid workflow Γ itself also forms a region. A node *N'* is thus said to be *within* a region $\{N_1, N_2\}$ (denote by $N' \subset \{N_1, N_2\}$) if there exists a path $N_1 \rightarrow \ldots \rightarrow N' \rightarrow \ldots \rightarrow N_2$. The definition of *region* preserves the following property.

Proposition 1: If $\{N_1, N_2\}$ and $\{N_2, N_3\}$ are two regions in Γ where N_1, N_2, N_3 are uniquely identified nodes, $\{N_1, N_3\}$ also forms a region in Γ.

To well control the number of regions that can be identified in a grid workflow and to make them cover the whole grid workflow, it is required to further define a standard granularity for regions.

Definition 2 (Maximized Region): Two *Srv&Ctrl* nodes or *Begin / End* nodes N_{head} and N_{tail} form a *maximized region* in a grid workflow Γ, IFF $\forall$ *Begin* $\rightarrow N_1 \rightarrow \ldots N_m \rightarrow$ *End* where *Begin*, *End* are the *Begin* node and the *End* node of Γ, N_{head} and N_{tail} are contained in the path and $N_{head} \neq N_{tail}$.

It can be seen from the definition that a maximized region is also a region, and thus satisfies Proposition 1.

Definition 3 (Decomposition): A maximized region $\{N_1, N_2\}$ in Γ is said to be *decomposable* IFF: (1) there are more than one path $N_1 \rightarrow \ldots \rightarrow N' \rightarrow \ldots \rightarrow N_2$ s.t. N_1 can reach N_2 in Γ and $\exists N' \subset \{N_1, N_2\}$, s.t. $\{N_1, N'\}$ or $\{N', N_2\}$ is a maximized region; or (2) there is only one path $N_1 \rightarrow \ldots \rightarrow N' \rightarrow \ldots \rightarrow N_2$ s.t. N_1 can reach N_2 in Γ and for any $N_0 \rightarrow N_1$ and $N_2 \rightarrow N_3$, either $\{N_0, N_2\}$, $\{N_1, N_3\}$ or $\{N_0, N_3\}$ is a maximized region. Moreover, for nodes $N'_1, \ldots N'_m$ within region $\{N_1, N_2\}$, the set of maximized regions $\{\{N', N''\} \mid \{N', N''\} = \{N_1, N'_1\}$ or $\{N'_1, N'_2\}$ or … or$\{N'_m, N_2\}\}$ is said to be a *total decomposition* of $\{N_1, N_2\}$ IFF all $\{N', N''\}$s are maximized regions and are not further decomposable.

Definition 4 (Standard Region): A maximized region $\{N_1, N_2\}$ in Γ is a *standard region* IFF $\{N_1, N_2\}$ belongs to the total decomposition of Γ.

Since Γ itself also forms a maximized region, Definitions 2 and 3 imply that a standard region will always exist for Γ (in the worst case the only standard region will be Γ itself). For example in Fig. 2, while {*TrigBank_H2_3, thIncaII_L1H2*} is a region, it is neither a maximized region nor a standard region.

However, {*Begin, Inspiral*} is a standard region for Γ. As the following proposition states, standard regions enjoy important sequential relations between each other that can be used for verification decomposition.

Proposition 2: For any two different standard regions $\{N_1', N_2'\}$ and $\{N_1'', N_2''\}$ in Γ, one of the following two temporal relations can be held:

1) Any *Srv&Ctrl* node N'' in $\{N_1'', N_2''\}$ (including N_1'' and N_2'') is *preceded* by any *Srv&Ctrl* node N' in $\{N_1', N_2'\}$ (including N_1' and N_2'), in that $N' \rightarrow \ldots \rightarrow N''$ always exists with zero or multiple transitions / data channels in all paths in Γ;

2) Any *Srv&Ctrl* node N' in $\{N_1', N_2'\}$ (including N_1' and N_2') is *preceded* by any *Srv&Ctrl* node N'' in $\{N_1'', N_2''\}$ (including N_1'' and N_2''), in that $N'' \rightarrow \ldots \rightarrow N'$ always exists with 0 or multiple transitions / data channels in all paths in Γ.

The proposition indicates that strict precedence relations are preserved between standard regions. Accordingly the corresponding algorithm (*TotalDecomposition*) for decomposing a grid workflow into its standard regions is also implemented in this work. Fig. 2 also shows the result of standard regions in the given case study of gravitational wave data analysis.

4 Verification Decomposition based on Standard Regions

Apart from the decomposition of grid workflows, the decomposition of corresponding formal verification strategies are also developed, which include:

1)How to exploit the properties of a standard region into its verification;

2)How to exploit local verification of a standard region into verification of other standard regions;

3)How to deduct the global verification result based on local verification of standard regions.

Above issues can be actually transformed into a special modular model checking problem [25]. As we know, the idea of formal verification is to find all states $\{s \in M \mid M, s \models f\}$, where M is the state model [13] (e.g. kripke structure, automata, etc) of the target system to be verified and f is the desired property. It is said that M satisfies f (i.e. $M \models f$) if the set of states s is not empty. A modular model checking tries to deduct the formal verification procedure in the following form:

$$\frac{<TRUE> M <\varphi> \qquad <\varphi> M' <\psi>}{<TRUE> M \mid M' <\psi>} \qquad (1)$$

The deduction tries to prove that if model M satisfies property φ ($<TRUE>M<\varphi>$) and model M' satisfies property ψ under the *assumption* that its environment satisfies property φ ($<\varphi>M'<\psi>$), the parallel composition of ($M|M'$) will satisfy property ψ ($<TRUE>M|M'<\psi>$). An essential procedure in the above deduction is how to define and implement $<\varphi>M'<\psi>$ such that the deduction will hold true. Consequently, our decomposition strategy of verifications based on standard regions follows the idea below: given the total decomposition $\{M_1, M_2, \ldots, M_n\}$ of a grid workflow Γ where $M_i = \{N_i, N_{i+1}\}$, $N_i, N_{i+1} \in \Gamma$, the verification of a desired property ψ is carried out on $M_n, \ldots, M_1$

separately, whereas the verification of M_i against ψ will be based on the satisfaction of $M_{i+1};\ldots;M_n$ against ψ such that the satisfaction of the complete workflow Γ against ψ can be eventually deducted.

$$\frac{<TRUE>M_{i+1};.....;M_n<\psi_{i+1}> \qquad <\psi_{i+1}>M_i<\psi_i>}{<TRUE>M_i;M_{i+1};.....;M_n<\psi_i>} \quad (2)$$

Here we have $1 \le i \le n-1$ and *M;M'* indicates the sequential composition of identified standard regions since sequential relations are preserved among standard regions. The following takes LTL-X (a popular temporal logic with universal path qualifiers and no *next* operators) [13] as the target for the implementation of $<\varphi>M'<\psi>$ (i.e. both φ and ψ are specified in LTL-X). LTL-X is an intuitive and shuttering closed logic with wide formal verification tool support. Since an important theoretical foundation is that LTL-X formulae can be transformed to an equivalent generalized büchi automata [13], $<\varphi>M'<\psi>$ can be obtained by verifiying $Trans(\varphi)|M' \models \psi$, where $Trans(\varphi)$ indicates the equivalent automata for φ. However in this work, the sequential nature of standard regions enables us to further avoid the cost for automata composition. The following provides the implementation of $<\varphi>M'<\psi>$ based on the semantics of LTL-X formulae on system states and paths.

Definition 5 (Association States): Given the total decomposition $\{M_1, M_2, \ldots, M_n\}$ of a grid workflow Γ where $M_i=\{N_i, N_{i+1}\}$, N_i, $N_{i+1} \in \Gamma$, denote $TransSys(\Gamma, \Phi)$ to be the automata for Γ under the given initial state set of Φ. Since M_i and M_{i+1} share the service node N_{i+1}, the set of *association states* $Im(M_i, M_{i+1}, \Gamma)$ is the states when $M_i;M_i;\ldots;M_n$ transits to the process of $M_{i+1};M_{i+2};\ldots;M_n$.

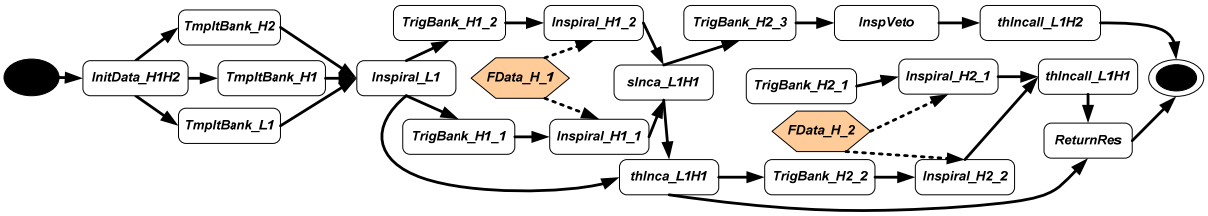


Fig. 2. Gravitational wave data analysis – case study I (*SF1*)

The association states literally indicate the *region initial states* for the previous local verification

$(<TRUE>M_{i+1};M_{i+2};\ldots;M_n<\psi_{i+1}>)$

and the *region ending states* for the current local verification $(<\psi_{i+1}>M_i<\psi_i>)$ in the deduction procedure (2).

Definition 6 (Region Initial / Ending States): Given the total decomposition $\{M_1, M_2, \ldots, M_n\}$ of a grid workflow Γ where $M_i=\{N_i, N_{i+1}\}$, $N_i, N_{i+1}\in\Gamma$, the *region ending states* for M_i ($\mathbb{E}(M_i)$) and the region initial states for $M_{i+1};M_{i+2};\ldots;M_n$ ($\mathbb{S}(M_{i+1};M_{i+2};\ldots;M_n)$) are both $Im(M_i, M_{i+1}, \Gamma)$.

Lemma: ∀ state

$s_i \in Trans(M_i, \mathbb{S}(M_i;\ldots;M_n))$ and

$s_{i+1} \in Trans(M_{i+1}, \mathbb{S}(M_{i+1};\ldots;M_n))$,

if $s_i = s_{i+1}$, then $s_i, s_{i+1} \in Im(M_i, M_{i+1}, \Gamma)$.

The lemma states that in the total decomposition of Γ, the only shared states of the corresponding automata for standard regions M_i and M_{i+1} are their association states. This implies that no states in one standard region will loop back to states in another standard region, which is a direct result of Definition 3 and Constraint 6.

Proposition 3: Given the total decomposition $\{M_1, M_2, \ldots, M_n\}$ of grid workflow Γ, the desired formula Ψ and all of its sub-formulae $\varphi\in sub(\Psi)$, based on the result that

$Trans(M_{i+1};M_{i+2};\ldots;M_n,\mathbb{S}(M_{i+1};\ldots;M_n))\models\varphi$

(i.e. $<TRUE>M_{i+1};M_{i+2};\ldots;M_n<\varphi>$), it can be deduced that

$<TRUE>M_i;M_{i+1};M_{i+2};\ldots;M_n<\varphi>$

also holds if:

- If $\varphi=p$ where p is an atomic Boolean proposition, $<\varphi>M_i<p>$ holds IFF $M_i, \mathbb{S}(M_i;\ldots;M_n)\models p$;
- If $\varphi=\varphi_1\vee\varphi_2$, $<\varphi>M_i<\varphi_1\vee\varphi_2>$

holds IFF M_i, $\mathbb{S}(M_i;\ldots;M_n)\models\varphi_1$ or M_i, $\mathbb{S}(M_i;\ldots;M_n)\models\varphi_2$;

- If $\varphi=\neg\varphi_1$, $<\varphi>M_i<\neg\varphi_1>$ holds IFF M_i, $\mathbb{S}(M_i;\ldots;M_n)\not\models\varphi_1$;
- If $\varphi=\varphi_1$ U φ_2, $<\varphi>M_i<\varphi_1$ U $\varphi_2>$ holds IFF for any state sequence starting from $\mathbb{S}(M_i;\ldots;M_n)$: $\pi=\mathbb{S}(M_i;\ldots;M_n),\sigma_1,\sigma_2,\ldots,\sigma_k$, it has:
  - o $\exists i\in N^+$, s.t. $\sigma_i\in\pi$ and $\sigma_i\in\mathbb{S}(M_{i+1};\ldots;M_n)$, $(M_{i+1};\ldots;M_n,\sigma_i)\models\varphi_1$ U φ_2. $\forall$ an infinite state sequence $\pi'\in Trans(\varphi_1$ U $\varphi_2)=\sigma^0,\sigma^1,\sigma^2,\ldots$, for each such σ_i and $\pi''=\mathbb{S}(M_i;\ldots;M_n)$, $\sigma_1,\ldots,\sigma_i,\sigma^1,\sigma^2,\ldots$, $(M_i,\pi'')\models\varphi_1$ U φ_2 holds true; Otherwise:
  - o $(M_i,\pi^*)\models\varphi_1$ U φ_2, where $\pi^*=\mathbb{S}(M_i;\ldots;M_n),\sigma_1,\sigma_2,\ldots\sigma_k,\sigma_k,\ldots$
- If $\varphi=$G φ_1, $<\varphi>M_i<$G $\varphi_1>$ holds IFF for any state sequence starting from $\mathbb{S}(M_i;\ldots;M_n)$: $\pi=\mathbb{S}(M_i;\ldots;M_n)$, $\sigma_1,\sigma_2,\ldots\sigma_k$, it has:
  - o $\exists i\in N^+$, s.t. $\sigma_i\in\pi$ and $\sigma_i\in\mathbb{S}(M_{i+1};\ldots;M_n)$, $(M_{i+1};\ldots;M_n,\sigma_i)\models$G φ_1. $\forall$ an infinite state sequence $\pi'\in Trans(\varphi_1$ U $\varphi_2)=\sigma^0,\sigma^1,\sigma^2,\ldots$, for each such σ_i and $\pi''=\mathbb{S}(M_i;\ldots;M_n)$, $\sigma_1,\ldots,\sigma_i,\sigma^1,\sigma^2,\ldots$, $(M_i,\pi'')\models$G φ_1 holds true; Otherwise:
  - o $(M_i,\pi^*)\models$G φ_1, where $\pi^*=\mathbb{S}(M_i;\ldots;M_n),\sigma_1,\sigma_2,\ldots\sigma_k,\sigma_k,\ldots$
- If $\varphi=\varphi_1$ W φ_2, $<\varphi>M_i<\varphi_1$ W $\varphi_2>$ holds IFF $<\varphi>M_i<$G $\varphi_1\vee(\varphi_1$ U $\varphi_2)>$;
- If $\varphi=\varphi_1$ R φ_2, $<\varphi>M_i<\varphi_1$ R $\varphi_2>$ holds IFF $<\varphi>M_i<$G $\varphi_2\vee(\varphi_2$ U $(\varphi\wedge\varphi_1))>$l
- If $\varphi=$F φ_1, $<\varphi>M_i<$F $\varphi_1>$ holds IFF $<\varphi>M_i<TRUE$ U $\varphi_1)>$.

Proposition 3 implies an important decomposition strategy for formal verification based on standard regions. That is, given a standard region M_i in a grid workflow Γ, the desired LTL-X formula Ψ and its sub formulae $\varphi\in sub(\Psi)$, if $<TRUE>M_{i+1};M_{i+2};\ldots;M_n<\varphi>$ holds, we can deduct the satisfaction of $<TRUE>M_i;M_{i+1};M_{i+2};\ldots;M_n<\varphi>$ by

investigating whether *(Trans(M_i, $\mathbb{S}(M_i;\ldots;M_n)$);Trans($\varphi$))*, $\mathbb{S}(M_i;\ldots;M_n) \models \varphi$ holds. Here "*;*" represents the sequential composition of *Trans(M_i, $\mathbb{S}(M_i;\ldots;M_n)$)* and *Trans($\varphi$)* in order to construct the paths of $\mathbb{S}(M_i;\ldots;M_n),\sigma_1,\ldots,\sigma_i,\sigma^1,\sigma^2,\ldots$

5 Grid Workflow Decomposition using Relaxed Region Analysis

In above sections, a complete method for formal verification of grid workflows by decomposing both the workflow and the corresponding verification strategy is formulated. However, one deficiency of the above workflow decomposition is that it has imposed strong constraints (see Section II) on grid workflow structure analysis, which sometimes limits the improvement of the verification performance because the identified standard regions not small enough. For example in the decomposition result in Fig. 2, the identified standard region {*Inspiral_L1*, *End*} can be still considered as a complex sub workflow compared to the complete grid workflow.

It is found that one of key factors in decomposing the verification of a grid workflow Γ is to assure that

$$\text{TransSys}(M_i,\mathbb{S}(M_i;\ldots;M_n)) \supseteq \mathbb{S}(M_{i+1};M_{i+2};\ldots;M_n)$$

s.t. the sequential composition of $M_i;\ldots;M_n$ will not loose complete behaviors in the original grid workflow. Under this condition, it is inspired to relax Constraint 2 in Section IIIA to allow multiple End nodes in Γ such that potential parallel branches can also be discovered in addition to the sequential standard regions.

Relaxation of Constraint 2: Each grid workflow has exactly one explicit Begin node and can be relaxed to allow multiple End nodes. New End nodes after relation are named secondary end nodes (V_{End}).

For a standard region, denote Mi / ($N_1 \rightarrow \ldots \rightarrow N_m$) ($N_j \in M_i$, $j=1,\ldots,m$) as the operation of removing a branch in Γ with corresponding grid workflow nodes, transitions and data channels. It is then expected to find more potential standard regions in a grid workflow by

temporarily removing a selected branch, and to make the verification decomposition result still work in this relaxed context.

Definition 7 (Parallel Branch): In the total decomposition $\{M_1, M_2, \ldots, M_n\}$ of a grid workflow Γ, a connected path $\mathbb{CP}=N_1 \rightarrow^* N_2 \rightarrow \ldots \rightarrow V_{End}$ in Mn is called a parallel branch, IFF: (1) $\rightarrow^*$ is a transition with no condition; (2) in the total decomposition $\{M'_1,M'_2,\ldots,M'_m\}$ of M_n / $(N_2 \rightarrow \ldots \rightarrow V_{End})$, if $N_1 \in M'_i$ (i=1,2,…m), $\forall$ i<j≤m, $\nexists$ path $N_{j_1} \rightarrow \ldots \rightarrow N_{jk}$ in M_j, s.t. $\exists N_{jk} \in N_2 \rightarrow \ldots \rightarrow V_{End}$; (3) $\nexists$ other path $N_0 \rightarrow^* N_1 \rightarrow N_2 \rightarrow \ldots \rightarrow V_{End}$ in M'_i s.t. it also satisfies condition (2).

The above definition of a parallel branch $\mathbb{CP} = N_1 \rightarrow^* N_2 \rightarrow \ldots \rightarrow V_{End}$ indicates a path that ends with V_{End}, such that a parallel composition relation holds between grid service nodes in $N_2 \rightarrow \ldots \rightarrow V_{End}$ and new discovered standard regions after node N_1 (i.e. there are no control / data constraints in service execution among them). In the total decomposition $\{M'_1,M'_2,\ldots,M'_m\}$ of $M_n/(N_2 \rightarrow \ldots \rightarrow V_{End})$, if $N_1 \in M'_i$, it is called that the parallel branch $\mathbb{CP}$ belongs to M'_i, denoted by M'_i ($\mathbb{CP}$). $\{M_1,M_2,\ldots,M'_1,M'_2,\ldots,M'_i\ (\mathbb{CP}),\ldots,M'_m\}$ is therefore called the relaxed total decomposition after the relaxation of $\mathbb{CP}=N_1 \rightarrow N_2 \rightarrow \ldots \rightarrow V_{End}$ for grid workflow Γ.

Denote $\mathbb{CP}'=N_2 \rightarrow \ldots \rightarrow V_{End}$, because $\mathbb{CP}'$ forms a parallel relation with all the rest standard regions $(M'_{i+1},\ldots, M'_m)$ when M'_i ($\mathbb{CP}$), we have $\mathrm{Trans}((M'_k,\ldots, M'_m|\mathbb{CP}')) \supseteq \mathrm{Trans}((M'_{k+1},\ldots, M'_m|\mathbb{CP}'))$ for any $i \le k < m$ under the same initial state *Init*. Therefore the verification under relaxed region analysis can also reuse the results in Section IV. The whole deduction procedure includes four steps, as shown in (3)

$$\begin{cases} \dfrac{<TRUE>(M'_{j+1};.....;M'_m)|\mathbb{CP}'<\psi'_{j+1}> \quad <\psi'_{j+1}>M'_j|\mathbb{CP}'<\psi'_j>}{<TRUE>(M'_j;M'_{j+1};.....;M'_m)|\mathbb{CP}<\psi_i>} & i<j\le m \\ \dfrac{<TRUE>(M'_{i+1};.....;M'_m)|\mathbb{CP}'<\psi'_{i+1}> \quad <\psi_{i+1}>M'_i(\mathbb{CP}')<\psi'_i>}{<TRUE>(M'_i;M'_{i+1};.....;M'_m)|\mathbb{CP}'<\psi_i>} & \\ \dfrac{<TRUE>M'_{l+1};.....;M'_m<\psi'_{l+1}> \quad <\psi'_{l+1}>M'_l<\psi'_l>}{<TRUE>M'_l;M'_{l+1};.....;M'_m<\psi'_l>} & 1\le l\le i-1 \\ \dfrac{<TRUE>M_{k+1};.....;M_{n-1};M'_1;.....;M'_m<\psi_{k+1}> \quad <\psi_{k+1}>M_k<\psi_k>}{<TRUE>M_k;M_{k+1};.....;M_{n-1};M'_1;.....;M'_m<\psi_k>} & 1\le k\le n \end{cases} \tag{3}$$

The former two in (3) represent cases of standard regions with consideration of parallel branches, while the latter two

are used to deal with normal situations described in Section IV. The RRA flow chart is illustrated in Fig. 3, which is based on the TotalDecomposition algorithm.

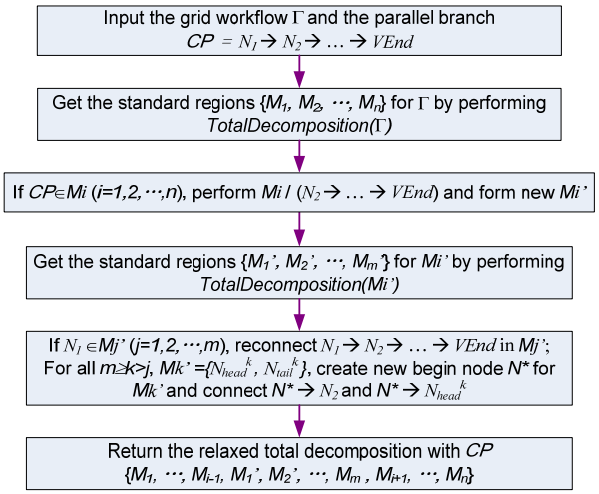


Fig. 3. The RRA flow chart

In the gravitational wave workflow SF1, since *sInca_L1H1* →* *TrigBank_H2_3* → *InspVeto* → *thIncaII_L1H2* → V_{End} is a parallel branch, the original standard region {*Inspiral_L1* , *End*} can be further decomposed into two smaller regions, {*Inspiral_L1*, *thInca_L1H1*} and {*thInca_L1H1*, *End*}, based on the RRA approach as described in Fig. 4.

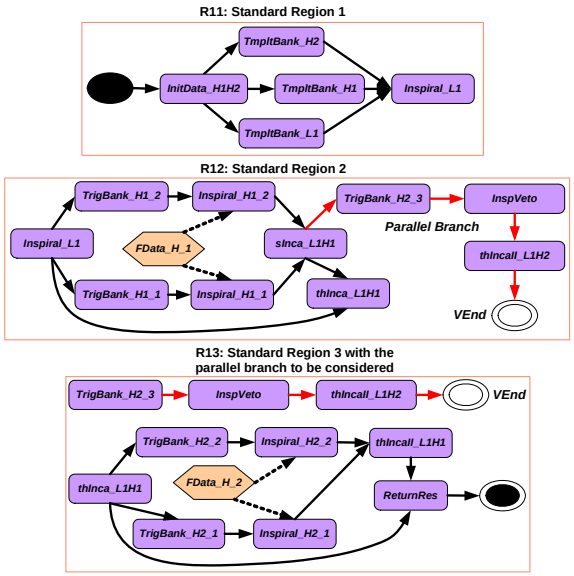


Fig. 4. Relaxed region decomposition for SF1

6 System Implementation and Performance Evaluation

6.1 GridPiAnalyzer

State Pi Calculus is used in our experiments for grid workflow modeling. The RRA approach with the verification strategy in Section III and IV is implemented in our GridPiAnalyzer to further improve its performance in reasoning grid workflows. GridPiAnalyzer is an automatic analyzer designed for ensuring reliability of grid workflow based on its Pi calculus formalism and verification.

GridPiAnalyzer accepts target grid

workflow scripts, e.g. DAG specifications, BPEL4WS and the Unified Modeling Language (UML) 2.0 activity diagrams. It automatically transforms grid workflow specifications into the process algebra of State Pi Calculus and deduces the result into labeled transition systems according to the operational semantics of State Pi Calculus. A visual environment is provided for specifying required temporal properties on grid workflows using LTL. These formulae, together with the transition system, are accepted to perform the formal verification.

GridPiAnalyzer is further extended with the capability of our RRA approach for performance improvement. A new component is developed to decompose grid workflows into standard regions with parallel branches based on the procedure described in Fig. 3. The formal verification is then recursively performed on each standard region instead of the whole grid workflow.

6.2 Case Studies

Let's take our scenarios of gravitational wave data analysis as case studies. Along with the relatively simple workflow *SF1* introduced in Fig. 2, two more complex ones *SF2* and *SF3* are also given in Figs. 5 and 6. We only focus on the temporal properties verifications on the grid scientific workflows, thus the detail of its functionalities are not introduced here, which could be referred to [3].

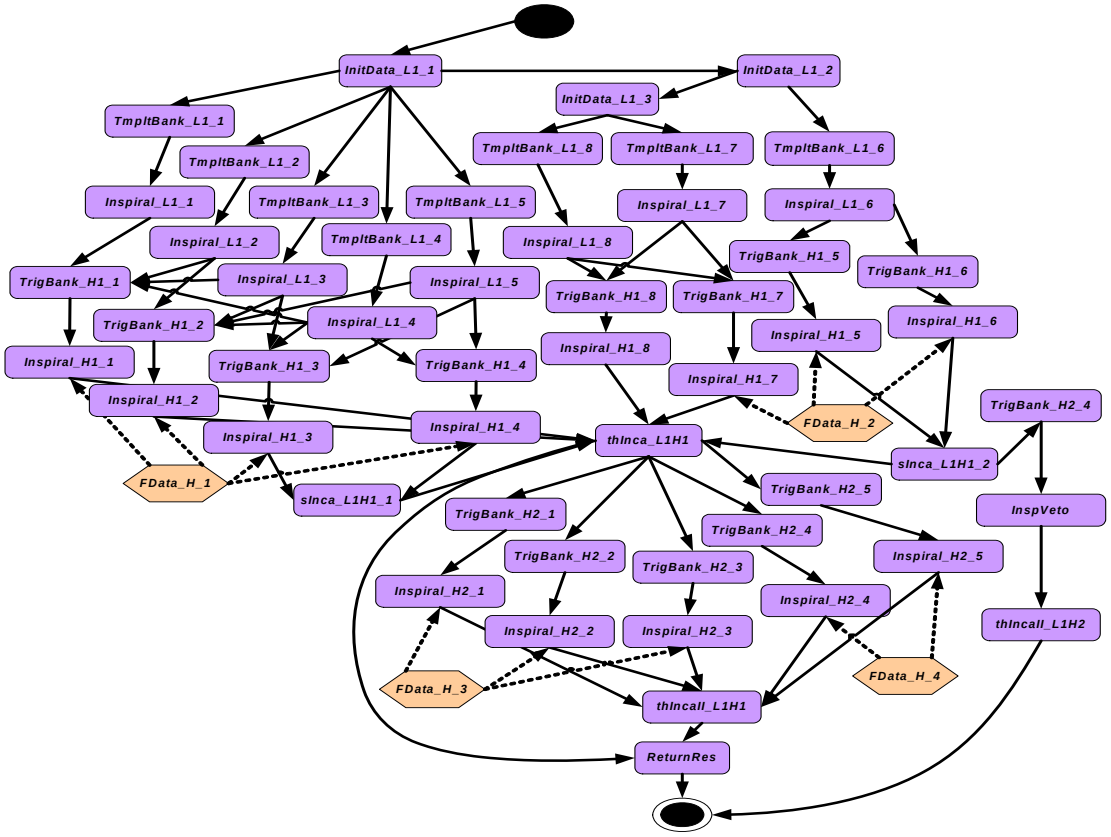


Fig. 5. Gravitational wave data analysis – case study II (*SF2*)

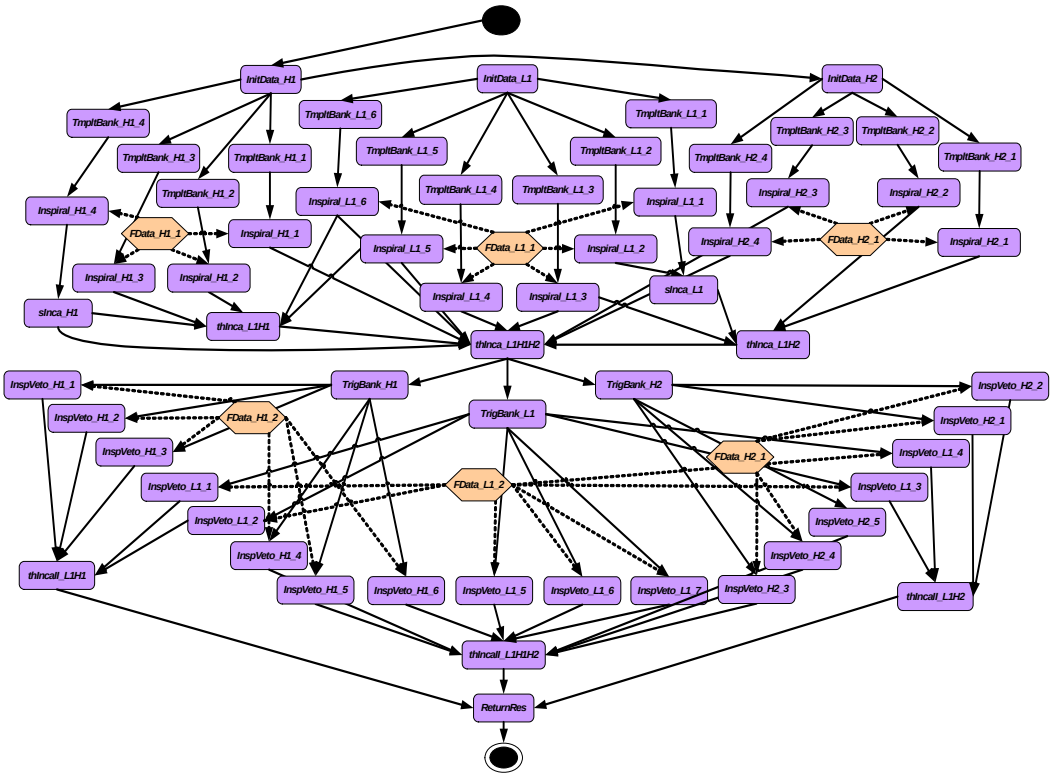


Fig. 6. Gravitational wave data analysis – case study III (*SF3*)

Corresponding relaxed standard regions are decomposed in Figs. 7 and 8, respectively.

- *p1*: The necessary successor operations after template bank generation;
- *p2*: The constraints on working status of laser interferometers;
- *p3*: The completeness of incidental analysis;
- *p4*: The precondition of final incidental analysis.

Correspondingly the LTL-X formulae of these properties for each grid workflow are formulated in Table I. These properties are required to be verified for all three case studies.

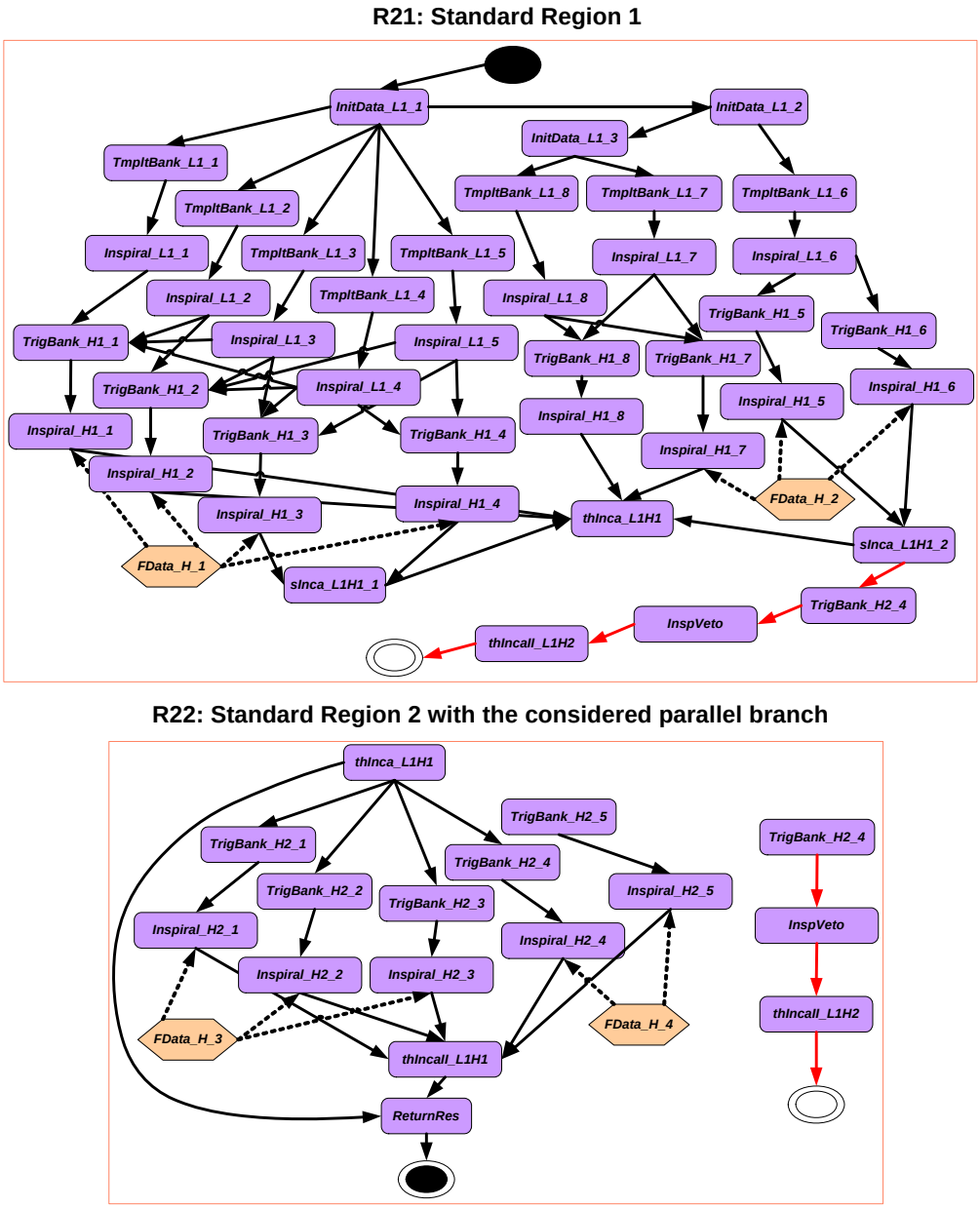


Fig. 7. Relaxed region decomposition for *SF2*

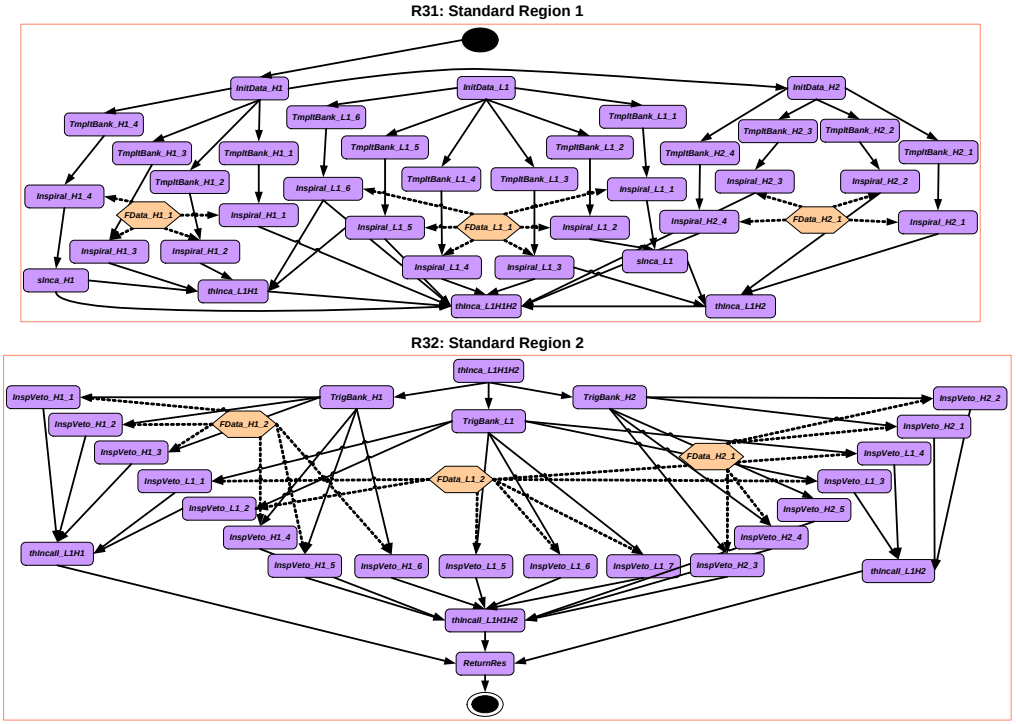


Fig. 8. Relaxed region decomposition for *SF3*

TABLE 1 FORMULAE OF PROPERTIES AGAINST *SF1*, *SF2* AND *SF3*

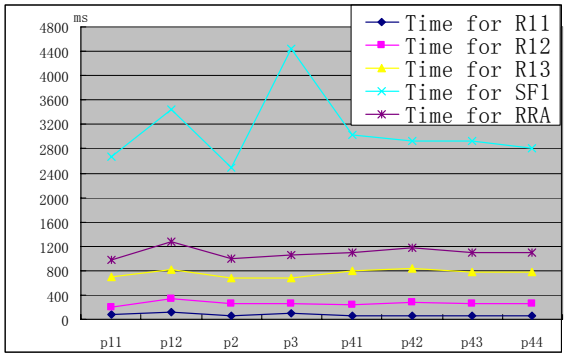
Prop Name	Target SF	Formulae for the Desired Property
$p1_1$	SF1 / SF3	G (*TmpltBank* _ *H*1 → ((F *TrigBank* _ *H*1) & (F *Inspiral* _ *H*1))
$p1_2$	SF1 / SF3	G (*TmpltBank* _ *H*2 → ((F *TrigBank* _ *H*2) & (F *Inspiral* _ *H*2))
p1	SF2	G (*TmpltBank* _ *L*1 → ((F *TrigBank* _ *H*1) & (F *Inspiral* _ *H*1))
p2	SF1 / SF2	G ((*InitData* _ *H*1*H*2) → (¬ *Inspiral* _ *H*2 U *thInca* _ *L*1*H*1))
	SF3	G ((*InitData* _ *H*1*H*2) → (¬ *Inspiral* _ *H*2 U *thInca* _ *L*1*H*1*H*2))
p3	SF1 / SF2	((F *sInca* _ *L*1*H*1 ∧ (¬*thIncaII* _ *L*1*H*1 U *sInca* _ *L*1*H*1)) ∨ (F *thInca* _ *L*1*H*1 ∧ (¬*thIncaII* _ *L*1*H*1 U *thInca* _ *L*1*H*1))) ∧ F *thIncaII* _ *L*1*H*1
	SF3	((F *sInca* _ *L*1 ∧ (¬*thIncaII* _ *L*1*H*1*H*2 U *sInca* _ *L*1) ∧ F *sInca* _ *H*1 ∧ (¬*thIncaII* _ *L*1*H*1*H*2 U *sInca* _ *H*1)) ∨ (F *thInca* _ *L*1*H*1*H*2 ∧ (¬*thIncaII* _ *L*1*H*1*H*2 U *thInca* _ *L*1*H*1*H*2))) ∧ F *thIncaII* _ *L*1*H*1*H*2
$p4_1$	SF1 / SF2	G (*Inspiral* _ *H*1 → (F *thIncaII* _ *L*1*H*1))
	SF3	G (*Inspiral* _ *H*1 → (F *thIncaII* _ *L*1*H*1*H*2))
$p4_2$	SF1 / SF2	G (*Inspiral* _ *H*2 → (F *thIncaII* _ *L*1*H*1))
	SF3	G (*Inspiral* _ *H*2 → (F *thIncaII* _ *L*1*H*1*H*2))
$p4_3$	SF1	G (*TmpltBank* _ *H*1 → (F *thIncaII* _ *L*1*H*1))
	SF2	G (*TmpltBank* _ *L*1 → (F *thIncaII* _ *L*1*H*1))
	SF3	G (*TmpltBank* _ *H*1 → (F *thIncaII* _ *L*1*H*1*H*2.))
$p4_4$	SF1	G (*TmpltBank* _ *H*2 → (F *thIncaII* _ *L*1*H*1))
	SF3	G (*TmpltBank* _ *H*2 → (F *thIncaII* _ *L*1*H*1*H*2))

6.3 Performance Evaluation

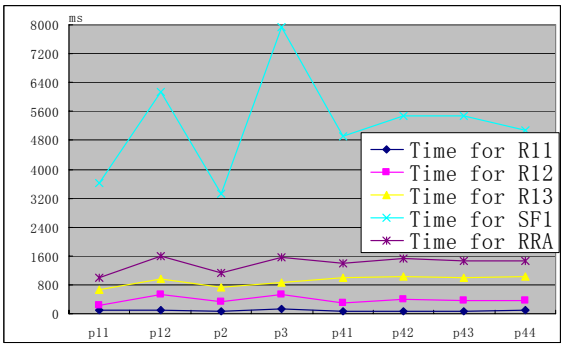
In this section, the proposed RRA

approach is applied in formal verification of grid workflows and its performance is compared with several well known verification methods [22]. These include the Symbolic Model Checking Algorithm (*SMCA*), *SMCA* with Cone of Influence (*COI*), *SMCA* with dynamic re-ordering of BDD (Binary Decision Diagram) variables (*Dynamic*), and Bounded Model Checking algorithm (*BMC*(*k*), where *k* is the setting of its length). All experiments are carried out using a machine with a Pentium IV 1.73GHz mobile CPU, 2.0GB RAM, the Windows operating system and Eclipse development platform.

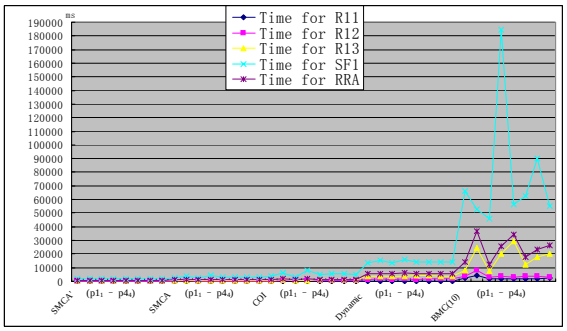
Performance evaluation results are illustrated in Figs. 9 and 10 in terms of verification time and peak memory usage, respectively.



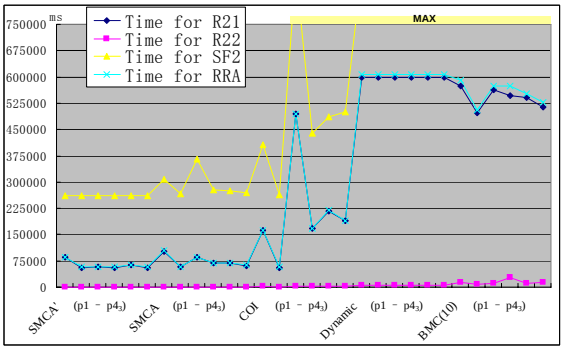
(a)



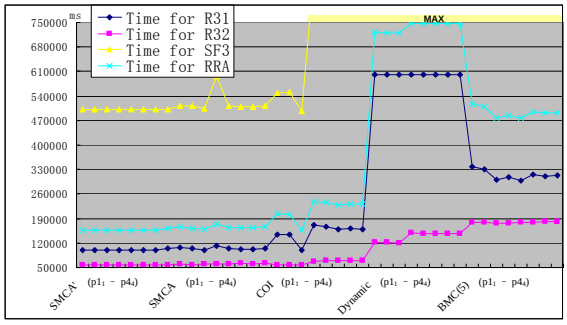
(b)



(c)

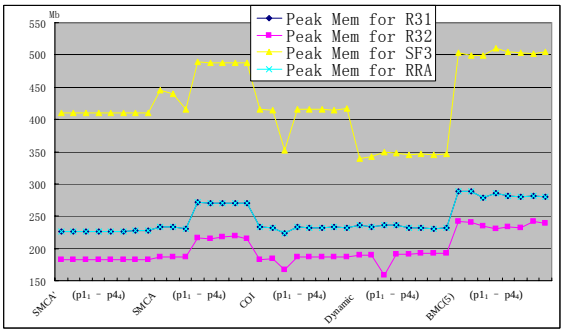


(d)



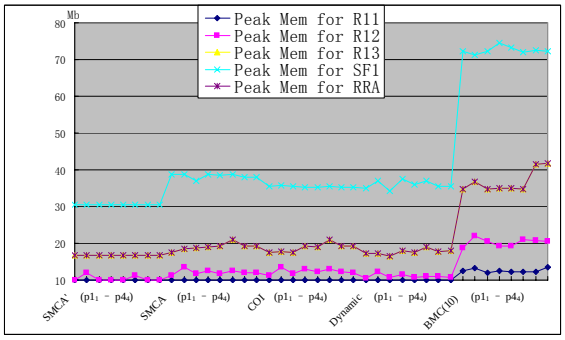
(e)

Fig. 9. Performance evaluation of verification time for *SF1*, *SF2* and *SF3*

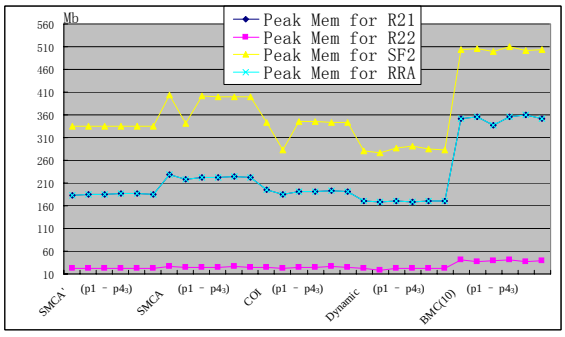


(c)

Fig. 10. Performance evaluation of perk memory usage for *SF1*, *SF2* and *SF3*



(a)



(b)

The upper limit of verification time is 750s for a complete grid workflow and 600s for standard regions. The minimum considered peak memory usage is 10Mb during grid workflow verification. Among legends in Figs. 9 and 10, *SMCA'* indicates the time / memory usage for the model initialization with the *SMCA* method. *SF1*, *SF2*, and *SF3* represent the three case studies illustrated in Figs. 2, 5 and 6, respectively. R_{ij} indicates the verification on the i^{th} identified standard region in SF_j. *RRA* indicates the verification result using the proposed approach. For a specific property *p*, the total verification time of *RRA*(t_p) is computed as follows: $t_p = \sum_{i=1}^{n} t_p(R_i)$ where $t_p(R_i)$ is the verification time for *p* on the i^{th} region in a grid

workflow; the peak memory usage of *RRA(m_p)* is computed as: $m_p = Max(m_p(R_i))$ where $m_p(R_i)$ is the peak memory usage for *p* on the i^{th} region in a grid workflow. The purpose of the additional Figs. 9(a) and 9(b) is to further zoom in the performance comparison with RRA and pure *SMCA* / *COI* approach in Fig. 9(c).

From the results included in Figs. 9 and 10, it can be found that due to the complexity of the grid workflow *SF2* and *SF3* (which contains 2^14.5 and 2^15.4 reachable states, respectively), direct verification of these workflows with none of the compared methods shows satisfactory performance. Based on *SMCA* and *COI*, verification time exceeds 250s and 500s respectively; peak memory usage exceeds 400Mb and 480Mb respectively. Based on *Dynamic* and *BMC*, verification time even exceeds the upper limit and peak memory assumption exceeds 290Mb, 510Mb, 349Mb and 511Mb, respectively. Poor performance of the *BMC* approach is partially due to the use of the simple Mini-SAT solver. Performance comparison between *BMC* and *SMCA* is out of the scope of this paper.

Based on our proposed RRA approach, the verification time is reduced to over 60s, 160s, 590s and 480s for *SF2* and *SF3* in comparison with the *SMCA*, *COI* and *BMC* method respectively. The peak memory usage is dramatically reduced to 230Mb, 196Mb, 171Mb, and 360Mb for *SF2* and 270Mb, 232Mb, 236Mb, 290Mb for *SF3*. Here for RRA with the *Dynamic* method, the verification time for the first standard region in *SF2* and *SF3* still exceeds the upper limit. This is because the *Dynamic* method is in essential a memory saving optimization technique which may worsen the verification efficiency.

By applying the RRA approach, memory usage savings are guaranteed since RRA enables partial loading and verification of grid workflows. Since state spaces of separate regions are also reduced compared to the global workflow, the proposed RRA approach not only reduces verification time, but also the time for BDD operation, Boolean satisfiability solving, memory operations, etc in the real implementation of *SMCA* and *BMC* approaches. These result in global performance improvement for the proposed RRA approach.

Conclusion

In this work, We propose State Pi Calculus, an extension of the existing Pi Calculus for formal representation of grid services. Formalism of different activities and control structures using State Pi Calculus is implemented to describe scientific workflows composed with grid services.

Furthermore, the decomposition strategy for standard regions based verification of grid workflows is proposed to enhance the performance in formal verification of grid workflow correctness. The RRA approach can effectively decompose a grid workflow into separate standard regions with parallel branches. Consequently, costly global reasoning of a grid workflow can be decomposed into light-weight local reasoning of its standard regions, each with a reduced state space.

The approach is implemented in our *GridPiAnalyzer*, an automatic formal verification system. Detailed experimental results show that verification overhead in terms of both CPU time and memory usage are dramatically reduced by applying our RRA approach, compared with using various traditional formal verification algorithms directly.

The disadvantages of this work is that it works on single workflow only, without considering more on multiple grid workflows that interact with each other. However this problem can be overcomed by a formal representation of multiple interactive workflows using state Pi calculus, with introducing more semantics regulations.

Ongoing work includes refinement of the *GridPiAnalyzer* system, with the implementation of more grid workflow verification modules, especially to interactive multiple workflow systems. Applying the RRA approach to more and complex real world applications is also our future research direction.

References

[1] Foster I and Kesselman C, *The Grid: Blueprint for a New Computing Infrastructure*, Morgan-Kaufmann, 1998.

[2] Cao J, Jarvis S A, Saini S, and Nudd G R, GridFlow: Workflow Management for Grid Computing, In *Proc. 3rd IEEE/ACM Int. Symp. on Cluster*

Computing and the Grid, Tokyo, Japan, May 12-15, 2003, pp. 198-205, 2003.

[3] Brown D A, Brady P R, et al, A Case Study on the Use of Workflow Technologies for Scientific Analysis: Gravitational Wave Data Analysis, in I. J. Taylor, D. Gannon, et al (Eds.), *Workflows for eScience: Scientific Workflows for Grids*, Springer Verlag, pp. 39-59, 2007.

[4] Cao J, Fingberg J, Berti G, and Schmidt J G, Implementation of Grid-enabled Medical Simulation Applications Using Workflow Techniques, in *Proc. 2nd Int. Workshop on Grid and Cooperative Computing*, Shanghai, China, Dec. 7-10, 2003, pp. 34-41, 2003.

[5] Liu Y, Müller S, and Xu K, A Static Compliance Checking Framework for Business Process Models, *IBM Systems Journal*, 2007, 46(2): pp. 335-362.

[6] Chen J and Yang Y, A Taxonomy of Grid Workflow Verification and Validation, *Concurrency and Computation: Practice and Experience*, 2008, 20(4): pp. 347-360.

[7] Chen J and Yang Y, Multiple States based Temporal Consistency for Dynamic Verification of Fixed-time Constraints in Grid Workflow Systems, *Concurrency and Computation: Practice and Experience*, 2007, 19(7): pp. 965-982.

[8] Tan W, Fan Y, Zhou M, A Petri Net-based Method for Compatibility Analysis and Composition of Web Services in Business Process Execution Language. *IEEE Transactions on Automation Science and Engineering*. 2009, 6(1): pp. 94-106

[9] Tan W, Fan Y, Zhou M, Tian Z, Data-driven Service Composition in Building SOA Solutions: A Petri Net Approach. *IEEE Transactions on Automation Science and Engineering*. 2010, 7(3): pp. 686 - 694

[10] Li X, Fan Y, Sheng Q Z, Maamar Z, Zhu H, A Petri Net Approach to Analyzing Behavioral Compatibility and Similarity of Web Services. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans* , to be online, 2010.

[11] Xiong P, Fan Y and Zhou M, Web Service Configuration under Multiple Quality-of-Service Attribute, *IEEE Transactions on Automation Science and Engineering*, April 2009, 6(2): pp. 311-321

[12] Xiong P, Fan Y and Zhou M, QoS-aware Web Service Configuration, *IEEE Transactions on System, Man and Cybernetics, Part A*, July 2008, 38(4): pp.888-895

[13] Clarke E M, Grumberg O, and Peled D A, *Model Checking*, MIT Press, 1999.

[14] Xu K, Wang Y X, Wu C, Formal Verification Technique for Grid Service Chain Model and its Application. *Science in China, Series F: Information Sciences*, 2007, 50(1): pp. 1-20

[15] Xu K, Cao J, Liu L, and Wu C, Performance Optimization of Temporal Reasoning for Grid Workflows Using Relaxed Region Analysis. In *Proc.* 22nd *IEEE Int. Conf. on Advanced Information Networking and Applications Workshops*, GinoWan, Okinawa, Japan, March 25-28, 2008 187-194.

[16] Salaun G, Bordeaux L, and Schaerf M, Describing and Reasoning on Web Services Using Process Algebra, In *Proc. IEEE Int. Conf. on Web Services*,

San Diego, USA, June 6-9, 2004, pp. 43-50.

[17] Nemeth Z and Sunderam V, Characterizing Grids: Attributes, Definitions, and Formalisms, *J. Grid Computing*, 2003, 1(1): pp. 9-23.

[18] Huang S, and Mulcahy J J, Software Reuse in the Evolution of an E-Commerce System: A Case Study. *International Journal of Computing & Information Technology (IJCIT)*, 2(1). pp: 1-15.

[19] Cai H, Scale-Free Web Services, *IEEE 2007 International Conference on Web Services*, Salt Lake City, Utah, USA, July 9-13, 2007.

[20] Milner R, *Communicating and Mobile Systems: the Pi Calculus*, Cambridge University Press, 1999.

[21] Wang S and Armstrong M P, A Quadtree Approach to Domain Decomposition for Spatial Interpolation in Grid Computing Environments, *Parallel Computing*, 2003, 29(10): pp. 1481-1504.

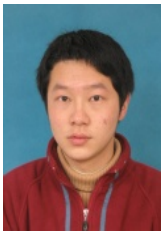
[22] Cimatti A, Clarke E et al, NuSMV2: an Open Source Tool for Symbolic Model Checking, *Computer Aided Verification*, LNCS 2002, Vol. 2404: pp. 359-364.

[23] Deelman E, Kesselman C et al, GriPhyN and LIGO, Building a Virtual Data Grid for Gravitational Wave Scientists, in *Proc. 11th IEEE Int. Symposium on High Performance Distributed Computing*, Edinburgh, Scotland, July 24-26, 2002, pp. 225-234, 2002.

[24] Liu R and Kumar A, An Analysis and Taxonomy of Unstructured Workflows, *Business Process Management*, 2005, Vol. 3469: pp. 268-284.

[25] Grumberg O and Long D E, Model Checking and Modular Verification *ACM Transactions on Programming Languages and Systems*, 1999 16(3): pp. 843-871.

Junwei Cao is currently Professor and Assistant Dean, Research Institute of Information Technology, Tsinghua University, China. He was a Research Scientist at MIT LIGO Laboratory and NEC Laboratories Europe. He received the PhD in Computer Science from University of Warwick, UK, in 2001. He is a Senior Member of the IEEE Computer Society and a Member of the ACM and CCF. Contact him via Email: jcao@tsinghua.edu.cn



Fan Zhang received the B.S. in computer science from Hubei Univ. of Technology and M. S. in control science and engineering from Huazhong University of Science and technology. He is currently a Ph.D. student in Department of Automation, Tsinghua University, Beijing, China. His research interests include data

center networks and grid/cloud computing. He is a 2010-2011 IBM Ph.d Fellow. He can be contacted via email at: zhang-fan07@mails.tsinghua.edu.cn

Ke Xu is an research engineerer in Morgan Stanley, Shanghai, China. He received his BSc Degree in 2002 from Automation Department of Shanghai JiaoTong University, PR China and PhD degree at the National CIMS Research and Engineering Center in Tsinghua University, China in 2007. His main research interests include grid computing, business integration, formal methods and model checking. He is also an IBM PhD. Fellow in year 2006–2007.

Lianchen Liu is currently an Associate Professor of Department of Automation, Tsinghua Univ. China. He received the Ph.D. from NanKai Univ., China. His research interests include large scale scientific resource sharing, distributed computing, etc. Contact him via Email: liulianchen@tsinghua.edu.cn

Cheng Wu is a Professor of Department of Automation, Tsinghua Univ. China, Director of National CIMS Engineering Research Center, and Member of Chinese Academy of Engineering. He received his B.S. and M.S. from Department of automation, Tsinghua Univ. in 1962 and 1966 respectively. His research interests include complex manufacturing system scheduling, grid/cloud applications, etc. Contact him via Email: wuc@tsinghua.edu.cn

扩展中文摘要：

网格技术的快速发展带来了网格应用复杂程度的不断提升。为了确保网格工作流执行期具有足够的可靠性和可信性，排查并验证在应用设计的初始阶段中的时序特性中的错误的问题在这种背景下尤为重要。一种状态Pi演算方法在本论文中提出，其不仅能够使能弹性化网格系统事件的抽象性和管理性，同时也能有助于网格工作流的建模和时序验证。同时，一种基于松弛域分析的方法也一并提出，用以将大规模网格工作流降解到低维度并行化且带有旁支工作流的域中，相应的验证策

略同时也自适应的降解。在引力波探测数据分析的真实试验中，验证了本方法能够大大降低验证过程的CPU和内存消耗量。

关键词：网格计算；工作流管理；形式化验证；状态Pi演算