

マルチエージェント開発環境RXFにおける リフレクションを用いた分散トレーサについて

副島 大和 大園 忠親 新谷 虎松
名古屋工業大学知能情報システム学科

1. はじめに

ソフトウェア工学の分野で、デバッグ[1]をはじめとする開発環境の研究が盛んに行われてきていて、一応の成果をあげている。しかし、マルチエージェントシステムのための開発環境は、近年研究がはじまったばかりである。本研究では、マルチエージェントシステム（以下MAS）の開発を支援するプログラミング環境を提案する。

筆者らはエージェント指向プログラミング（以下AOP）環境としてRXF[2]を開発した。本論文ではRXFの分散トレーサについて主に述べる。本論文では分散トレーサによって効果的なMASのデバッグを行えることを示す。

2. RXFとリフレクション

RXFは、AOPを効果的に実現するためにマルチスレッド、リフレクションなどの機構を実装した、Prologベースの論理型言語である。RXFのリフレクションは図1のような階層化されたスレッド機構によってリフレクティブタワー[4]を構成することで実現される。メタレベルのスレッドがベースレベルのスレッドのメタ表現を持ち、メタ表現を改変することによってベースレベルのスレッドの状態を改変することをメタレベル実行と呼ぶことにする。RXFではメタレベル実行を行うことによって自己改変や自己参照といったリフレクションの機能を実装している。リフレクティブタワーの階層数は理論上無制限であるが、本実装では、物理的メモリの容量の限りである。

3. トレーサの概略

既存の論理型言語での開発サポート技術として、boxモデルに基づくトレーサ[3]がある。このトレーサは、シングルスレッド実行のデバッグでは非常に効果的である。MASは複数のエージェントがネットワーク上に分散して複数のスレッドを並行動作させている。従って、本研究では分散してマルチスレッド実行をする論理型言語プログラムに対応したboxモデルに基づくトレーサを拡張したトレーサの実装を提案する。複数のエージェントのマルチスレッドのトレーサによるデバッグ情報は膨大なものであり、プログラマ

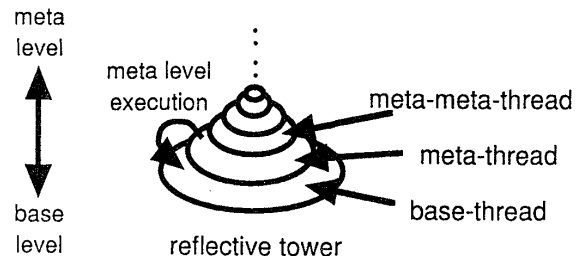


図1 RXFにおけるリフレクション

は膨大な情報を観察することは非常に困難である。そこで、デバッグの際に複数のエージェントを並列に同期をとって（後述）、動作させる機能（以下同期実行）を実現した。この機能によってプログラマは分散並列して動作するMASの観察を容易におこなうことが可能となる。また、エージェントウィンドウシステムという仕組みを導入することによって、膨大な量になるMASのデバッグ情報をプログラマが容易に整理できるようにした。

ネットワーク上の遠隔地で動作する可能性のあるエージェントをデバッグするには、エージェントをネットワーク透過に扱えるようにするための機能が不可欠である。そこで、エージェントの存在していないマシンからも分散してトレースする仕組みを実装した。これはデバッグによるハイゼンベルク効果（Heisenberg effect）[1]を軽減するためでもある。

リフレクションを利用したfix-and-run（修正して実行）[1]をサポートする機構も実装した。この機能は再実行時の時間的ロスの軽減と共に、分散並列計算でしばしば起きる再現性乏しい挙動におけるバグを発見時に修正できるようにするための仕組みである。

4. リフレクションを用いたトレーサと制御

トレーサからエージェントをデバッグできるように、図2のようにデバッグ対象の各エージェントには最もメタなレベルのスレッドのさらに一つ上の階層にメタレベルスレッド（以下デバッグスレッド）を作るようにしている。デバッグスレッドをトレーサから制御することで、トレーサはデバッグ対象のエージェントを自由に操作可能である。

同期実行は、図2のようにsuspend命令をデバッグ対象のすべてのエージェントのデバッグスレッドにトレーサから送るようにすることで実現する。トレーサからsuspendを送る間隔を調整することで同期実行のスピードを制御することもできる。

fix-and-runを実現するために、RXFのリフレク

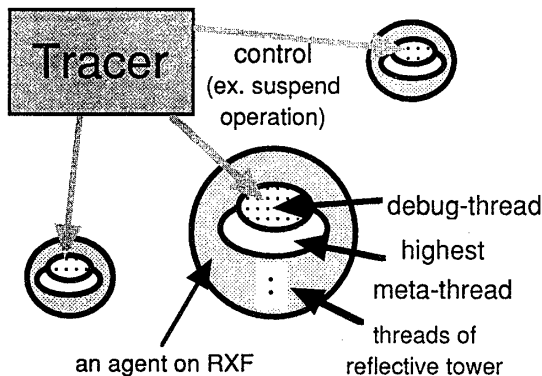


図2 トレーサによるエージェントの制御方法

ションの自己参照と自己改変の機能を利用している。この機能の実装により、プログラマはデバッグ中にバグを発見した際は、直ちに同期実行を止めて、その場でプログラムを修正してそのまま再実行できる。また、ある実行状態において、ある変数束縛のデータを改変して再実行することも可能である。

5. 実行例

図3に本トレーサの実行例を示す。図3上のウィンドウがエージェントウィンドウであり、分散トレーサのユーザインターフェースである。このウィンドウの③のテキストエリアからはRXFのQuery実行も可能である。複数のマシンで同時に一つのエージェントを観察することも可能である。エージェントウィンドウのインターフェースではエージェントのなかで並行動作している複数のスレッドの一覧を①のチョイスボタンに表示可能でそこからプログラマが目指すスレッドのトレースのみを表示可能である。②はboxモデルで言う深さを制限してトレースログを表示するものである。図3下のウィンドウが同期実行を制御する部分である。④は全てのエージェントに同期をとって実行状態を変化させるようにするための制御用ボタンである。左から順に、実行の開始、stop、suspend、sleep、resumeのボタンである。⑤は同期実行時にどの程度の間隔でsuspend命令をトレーサからエージェントに送信するかを決めるスライダーである。このバーによりみかけ上の実行スピードを調整できる。⑥はエージェントウィンドウを作るための一覧（リスト）とボタンである。プログラマが目指すエージェントのエージェントウィンドウのみを表示することによって、過剰な情報をユーザに表示しないようにする仕組みである。トレーシングはこれら二種のウィンドウを用いて行い、バグ発見時にはエディタを用いて修正し再実行する。

6. おわりに

本研究ではRXFにおける分散トレーサを実装した。

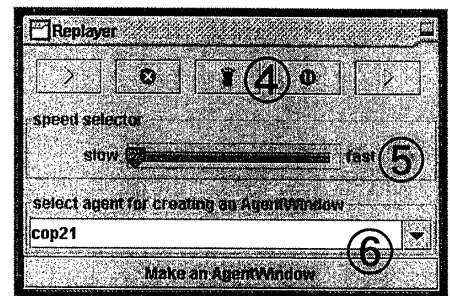


図3 分散トレーサの実行例

分散トレーサによって、分散並列して動作するMASの実行トレースをプログラマに従来のシングルスレッド用のトレーサよりも効果的に提示することが可能となった。また、分散してトレースすることによってハイゼンベルク効果の軽減と、ネットワーク透過にデバッグする仕組みを実装した。fix-and-runの機能によって、再実行時の無駄な時間を省くとともに、再現性の乏しいバグのデバッグも行えるようにしている。

分散トレーサの機能によって、プログラマのMASの開発の負担軽減に繋がったと考えられる。

今後の課題としては、同期実行以外のときのハイゼンベルク効果の軽減と、MASのデバッグ情報の表示方法のさらなる改善がある。

参考文献

- [1]Jonathan B. Rosenberg : How Debuggers Work, John Wiley & Sons, Inc.(1997)
- [2]大園忠親, 新谷虎松: 自律的エージェントのための制約論理型言語RXFにおけるリフレクション機構の設計とその実装, 情報処理学会論文誌, Vol.38, No.7, pp.1361-1369 (1997) .
- [3]Byrd, L. : Understanding the Control Flow of Prolog Programs, Proc. of the Logic Programing Workshop (1980)
- [4] 渡部卓雄: リフレクション, コンピュータソフトウェア, Vol. 11, No. 3, pp. 5-14 (1994) .
- [5]副島大和, 大園忠親, 新谷虎松: エージェント記述言語RXFにおけるエージェントプログラミング支援機構について, 人工知能学会全国大会 (第12回) 論文集, 人工知能学会, pp.203-204, (1998) .