水産物加工向け 3 次元計測システムへの CSP 適用事例

高瀬竜一[†1]　菊野博昭[†2]
西卓郎[†1]　吉見隆[†1]　河井良浩[†1]

概要：並行処理の問題解決に，CSP を適用した事例を述べる．我々は，魚フィレの 3 次元形状を計測するアプリケーションの開発にモデル検査を取り入れた．きっかけは，マルチスレッドのあらゆる状態を調べる必要に迫られたことである．このシステムは画像処理の並行化によって計測時間を短縮した．同時に，データを効率よく受け渡すために共有メモリが必要となった．それによって直面した問題とモデル検査による解決，状態爆発と過度な抽象化を回避するための設計上の選択，スレッドの単体テストへの振る舞い仕様の活用と，本件から得られた知見について述べる．

1. はじめに

本稿は，産官の共同研究における形式手法の適用事例を紹介する．我々はベルトコンベヤで運ばれる魚フィレの，3 次元形状を計測するシステムを開発した（図 1）．得られた 3 次元データは，切り身加工システムへの入力に用いられる．加工の品質を向上させるため，魚フィレの下側も含めた全体形状を計測した．さらに画像処理を並行化し，計測の所要時間短縮を図った．本稿はその開発において直面した問題と対策について，モデル検査を中心に述べる．最後に本件から得られた知見をまとめる．

図 1　3D インテリジェンスポーションカッターType F
（株式会社ニッコー［釧路］）

2. 問題の背景

以下の経緯によって，形式手法の導入に至った．

本システムは魚フィレを切り身に加工するため，3 次元形状を計測する．計測には 4 組のステレオカメラを採用した[1]．画像処理は時間がかかるので，マルチスレッドで所要時間を短縮した．しかし画像はサイズが大きいので，スレッド間通信のためにコピーを繰り返すと性能が低下する．そこで Hyper Frame Vision [2]に準じて，大容量のフレームバッファを共有メモリとした．すなわち撮影用のフレームバッファを，並行化された画像処理のメッセージキューに併用する．スレッド間で画像を値渡しせず，参照を受け渡す，共有メモリによる非同期処理である．これは連続撮影に対する，画像処理の一時的な遅れも回復できる．一方でこの方式は，排他制御を誤ると容易にデッドロックする．

課題は排他制御の設計手段である．盲目的に排他制御し，クリティカルセクションを減らしても，デッドロックを回避できる保証はない．マルチスレッドが取り得る状態は膨大なので，テストとデバッグに頼る方法は障害の再現や調査が困難となる．その上，個々のスレッドの修正であっても全体的な振る舞いが変わってしまうため，障害の要因を一度に全て取り除かなければならない．

3. 仕様記述言語とツール

初期の実装は，フレームバッファに生じたレースコンディションを取り除くことができなかった．そこで再設計に着手し，形式手法よる根本的な解決を図った．まずスレッドの振る舞いを CSP(Communicating Sequential Processes)[3]で記号化した．Hyper Frame Vision は画像処理のアーキテクチャであり，前述の問題に対する解法は与えない．体系化された仕様記述言語が必要であった．

CSP の選定理由はツールが大きい．PAT Pro[4]は CSP のためのモデル検査と状態遷移の可視化機能を備える．これはモデルのすみやかな修正と検証に有用である．記述を並行処理に絞り込み，画像処理を除外することで，モデルを単純化できる．しかし画像処理の戦略は，しばしば並行処理の設計を左右する．より優れた計測手法の追求は試行錯誤であったので，モデリングの作業効率を重視した．

4. 設計上の選択

再設計にあたり，モデル検査を前提とするいくつかの選択を行った．モデルは，状態数の組み合わせ爆発を起こし

†1 国立研究開発法人 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology
†2 株式会社ニッコー
Nikko Corporation

© 2015 Information Processing Society of Japan

てはならない．かつ過度の抽象化で，実際の問題から乖離してはならない．そこで次の 2 点に取り組んだ．

- 同期による状態数の削減
- 同期プリミティブの実装

一つ目は状態爆発への対処である．同期は遅い方のスレッドに歩調を合わせることになる．しかし同期の追加で状態数を削減できるなら，性能よりモデル検査を可能にすることを優先した．例えば，並行化された画像処理は一連のステップを終えてから同期するよりも，各ステップで同期する方が状態数は少なくできる．

二つ目は抽象化の削減である．複数のスレッド間のマルチランデブ[5]を実装し，CSP のイベントに対応づけた．これにより他の同期プリミティブを近似する必要がなくなり，モデルからコードへの直接的な具体化が可能となる．

以上により，実装に近いモデルとモデル検査を両立した．モデルはデッドロックの他にライブロックの可能性も示した．魚フィレはコンベヤで運ばれるので，撮影するスレッドは最優先される．そのため本システムに公平性はない．これらに対し安全性と生存性をモデル検査で証明した．その探索の深さは無制限とした．その後，モデルを C 言語でコード化した．この変換は自動ではなく人手によった．

5. 単体テストのためのログ機能

トレース（イベントの実行列）にもとづいて，スレッドの単体テストを実施した．単体テストの実行時にイベントをログに記録し，終了時に出力すれば，ログに残されたトレースで異常がないかのチェックが可能となる．

このテストを実現するため，マルチランデブにログ機能を実装した．ログは全てのマルチランデブで共有し，同期するとき対応するイベントの名前を書き込む．ここで新たな排他制御を持ち込まないよう，ログは CAS 命令による非ブロッキングデータ構造とした[6]．

並行合成すると膨大でも，個々のスレッドなら状態を網羅できる．まず，各スレッドには十分なテストケースを作成する．テストケースごとに一つのトレースとすれば，振る舞いから外部選択の非決定性を排除できる．テストプログラムを，テスト対象とテストケースの二つのスレッドで構成する．そしてマルチランデブを 2 スレッドのランデブになるよう動作を切り替えて，単体テストを行った．

6. モデリングとテストの知見

関心事の分離によるモデルの単純化は，部分問題への分割とは異なり，ある種の射影となった．モデル化した並行処理と除外した画像処理は，相互に影響し合う．その影響は，両者が結びつく実装において顕在化する．画像処理の戦略は，統合テストを通じて振る舞い仕様に反映した．

しかし冒頭のデッドロックやレースコンディションはこの方法でテストできない．図 2 のグラフは，デッドロックの検査でツールが探索した状態数である．仕様は番号順に詳細化した．仕様 1 はフレームバッファの容量を考慮せず，仕様 2 は満杯／空／その他の三つに抽象化し，仕様 3 では容量を 8 に具体化した．実装はさらに大容量である．このマルチスレッドの状態の多さがテストを至難にする．

見出されたことは，実装よりもむしろ単体テストを，モデルに対応付ける必要性である．スレッドの単体テストは振る舞いが決定的になる．かつ少数でスレッドの振る舞いを網羅できる．そこで単体テストを重点的に実施した．そのための模索がマルチランデブの実装と，実行時のイベントをログに残して確認する枠組みとなった．

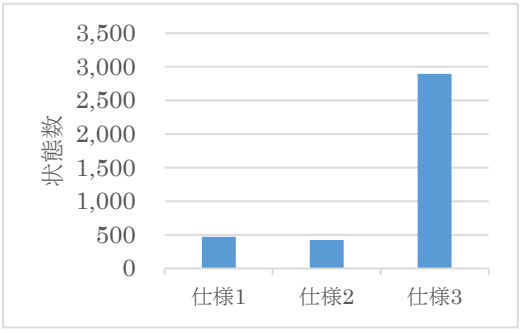

図 2　デッドロック検査で探索した状態数

7. おわりに

3 次元計測システムの研究開発に，CSP によるモデル検査を導入した．手法の有効性は発揮されたが，並行処理に限られた．一方で，カメラの高解像度化が進んでいるため，画像処理の異常追跡においてもデータ数の多さが難点となっている．今後は画像処理自体の問題について，形式手法をどのように応用可能かを検討する．

参考文献

1) 河井良浩，高瀬竜一，西卓郎，吉見隆，富田文明，菊野博昭: 水産物加工向け全周 3 次元形状計測システム．情報処理学会研究報告. Vol.2014-CVIM-191, No.32, pp.1-7 (2014).
2) Sumi, Y., Ishiyama, Y., Tomita, F.: Robot-Vision Architecture for Real-Time 6-DOF Object Localization. Computer Vision and Image Understanding, Vol.105, No.3, pp.218-230 (2007).
3) 磯部祥尚，東野輝夫: 並行システムの検証と実装．近代科学社 (2012).
4) Sun, J., Liu, Y., Dong, J.S.: Model Checking CSP Revisited: Introducing a Process Analysis Toolkit. The Third International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2008), pp.307-322 (2008). http://pat.sce.ntu.edu.sg/
5) Beton, R.: libcsp - a Binding Mechanism for CSP Communication and Synchronisation in Multithreaded C Programs. Communicating Process Architectures, Proceedings of WoTUG 23, Vol.58 of Concurrent Systems Engineering. pp.239-250 (2000).
6) 角川裕次: C とアセンブリ言語で学ぶ計算機プログラミングの基礎概念．森北出版 (2008).

© 2015 Information Processing Society of Japan