6
Conclusão

Finalizadores e referências fracas recebem suporte de praticamente todas as linguagens de programação que oferecem coleta de lixo automática. Para uma série recorrente de problemas, esses dispositivos podem ser usados em soluções elegantes, eficazes e por vezes únicas. Porém, ao contrário de finalizadores, referências fracas ainda são pouco conhecidas e pouco utilizadas, tanto na comunidade acadêmica quanto na indústria.

Referências fracas são um mecanismo mais simples e mais expressivo que finalizadores. Vimos que, quando referências fracas são associadas a um mecanismo de notificação, elas podem ser empregadas inclusive como um mecanismo alternativo de finalização. A finalização através de callbacks, via notificação ativa, possui algumas vantagens em relação aos finalizadores tradicionais:

- Quando o objeto finalizável é desacoplado da rotina de finalização, os atrasos na reciclagem de memória são evitados.
- Quando o objeto finalizável é passado como parâmetro do callback, ele continua acessível antes de ser efetivamente removido, fornecendo à aplicação um maior controle sobre a coleta.
- Callbacks podem ser associados a qualquer tipo da linguagem.
- Vários callbacks pode ser associados a um mesmo objeto, assim como um mesmo callback pode ser associado a vários objetos.

A notificação passiva fornece esses mesmo benefícios, porém, apresenta algumas vantagens em relação à callbacks e finalizadores. Conforme discutimos, em sistemas que empregam coletores de lixo baseados em rastreamento a utilização de callbacks ou finalizadores pode introduzir linhas de execução concorrentes na aplicação. Além disso, o indeterminismo desses coletores pode afetar negativamente o desempenho de algumas aplicações. Na notificação passiva, o problema de concorrência e sincronização é eliminado. Quanto à questão do indeterminismo, o coletor ainda é responsável por preencher a fila de notificações. O programa pode esperar por condições específicas para executar as ações associadas à finalização de um objeto, porém, isso irá depender do

coletor já ter inserido o objeto na fila. Por isso dizemos que a notificação passiva atenua o problema do indeterminismo, mas não o elimina completamente. Apesar de perder um pouco de automação, pois o programa deve invocar as rotinas de finalização explicitamente, acreditamos que as vantagens do mecanismo de notificação passiva o tornam uma opção mais adequada para a implementação de finalizadores na maioria dos casos.

Para todos os usos encontrados de finalizadores, pudemos elaborar uma solução através de referências fracas, mais especificamente, via um mecanismo de notificação passiva. Em alguns casos, esse mecanismo constituiu-se em uma solução mais simples e intuitiva. Baseado na discussão sobre finalizadores tradicionais, callbacks e filas de notificações apresentada no Capítulo 3, decidimos implementar um mecanismo de notificação passiva para Lua. Na implementação atual da linguagem, finalizadores podem ser usados apenas com um tipo específico, `userdata`. Com nosso mecanismo de notificação passiva, finalizadores podem ser usados com qualquer tipo da linguagem Lua. Além disso, eliminamos o problema de objetos ressucitáveis, pois o objeto permanece acessível até ser finalizado. No entanto, vimos que nossa implementação, como se encontra atualmente, não pode ser utilizada numa versão futura da linguagem Lua, pois a construção da fila de notificação não é devidamente controlada pelo coletor, o que pode ocasionar erros de memória.

Além da implementação de finalizadores via referências fracas, tratamos outro problema que constitui uma das contribuições mais importantes deste trabalho: a modificação do coletor de lixo da linguagem Lua para que este oferecesse suporte ao mecanismo de ephemerons. Agora, o programador pode optar por usar uma tabela de ephemerons ao invés de uma tabela fraca. Dessa forma, os ciclos existentes entre chaves e valores de uma tabela de ephemerons serão coletados. Isso torna essas tabelas uma opção ainda mais adequada que tabelas fracas para implementação de tabelas de propriedades. Tudo o que pode ser feito via referência fracas pode ser feito também via as tabelas de ephemerons. A fim de analisar a implementação do mecanismo de ephemerons, executamos testes de eficiência e realizamos uma análise informal do custo da coleta para tabelas e ephemerons e tabelas fracas. Como principal resultado, vimos que, na ausência de ciclos, tanto a coleta de tabelas fracas quanto a coleta de tabelas de ephemerons possuem o mesmo custo e o mesmo nível de eficiência. Contudo, quando existem ciclos, vimos que a eficiência da coleta de tabelas de ephemerons é bastante afetada. De fato, o custo passa de linear, no melhor caso (sem ciclos), para um custo exponencial, no pior caso. No entanto, devemos levar em consideração que a ocorrência do pior caso é rara. Sendo assim, devido a esses resultados, nossa implementação do mecanismo de

ephemerons para Lua pode ser incluída numa versão futura da linguagem.

6.1 Contribuições

Neste trabalho, tentamos mostrar como é possível implementar finalizadores via referências fracas. Além disso, resolvemos o problema de ciclos em tabelas fracas presente na linguagem Lua através do mecanismo de ephemerons. Resumidamente, as principais contribuições deste trabalho são:

- Efetuamos uma pesquisa informal sobre os usos de referências fracas e finalizadores na comunidade acadêmica e na indústria.
- Através do resultado da pesquisa e de uma pesquisa bibliográfica, identificamos e descrevemos os principais usos desses mecanismos.
- Mostramos, para cada uso encontrado de finalizadores, uma implementação através de referências fracas.
- Discutimos como o mecanismo de notificação passiva pode ser vantajoso em relação a callbacks e finalizadores tradicionais.
- Implementamos um mecanismo de notificação passiva para a linguagem Lua acoplado às tabelas fracas.
- Estudamos em detalhes o coletor de lixo de Lua e o mecanismo de ephemerons e estabelecemos a melhor adaptação desse último ao primeiro.
- Implementamos um suporte a ephemerons para a linguagem Lua, resolvendo o problema de ciclos em tabelas fracas. O programador pode agora utilizar uma tabela de ephemerons ao invés de uma tabela fraca tradicional.

Um ponto importante que não abordamos neste trabalho refere-se a como implementar um mecanismo de notificação passiva para Lua de forma mais adequada. Como foi discutido, o mecanismo que implementamos pode causar erros por falta de memória, já que na fase atômica da coleta de lixo, o próprio coletor precisa alocar memória para construir a fila de notificações e essa construção não é controlada adequadamente. Assim, como uma linha futura do trabalho, seria interessante estudar uma melhor implementação do mecanismo de notificação passiva.