[3] Pek E, Qiu X, Madhusudan P. Natural proofs for data structure manipulation in C using separation logic. In: Proc. of the 35th ACM SIGPLAN Conf. on Programming Language Design and Implementation. 2014,49(6):440−451. [doi: 10.1145/2594291.2594325]

[4] Distefano D, O'hearn PW, Yang H. A local shape analysis based on separation logic. In: Hermanns H, ed. Proc. of the Tools and Algorithms for the Construction and Analysis of Systems. Vienna: Springer-Verlag, 2006. 287−302. [doi: 10.1007/11691372_19]

[5] Lev-Ami T, Immerman N, Reps T, Sagiv M, Srivastava S, Yorsh G. Simulating reachability using first-order logic with applications to verification of linked data structures. In: Nieuwenhuis R, ed. Proc. of the Automated Deduction (CADE-20). Tallinn: Springer-Verlag, 2005. 99−115. [doi: 10.1007/11532231_8]

[6] Lev-Ami T, Sagiv M. TVLA: A system for implementing static analyses. In: Palsberg J, ed. Proc. of the Static Analysis. Santa Barbara: Springer-Verlag, 2000. 280−301. [doi: 10.1007/978-3-540-45099-3_15]

[7] Balaban I, Pnueli A, Zuck LD. Shape analysis by predicate abstraction. In: Cousot R, ed. Proc. of the Verification, Model Checking, and Abstract Interpretation. Paris: Springer-Verlag, 2005. 164−180. [doi: 10.1007/978-3-540-30579-8_12]

[8] Chatterjee S, Lahiri SK, Qadeer S, Rakamarić Z. A reachability predicate for analyzing low-level software. In: Grumberg O, ed. Proc. of the Tools and Algorithms for the Construction and Analysis of Systems. Braga: Springer-Verlag, 2007. 19−33. [doi: 10.1007/978-3-540-71209-1_4]

[9] Lahiri SK, Qadeer S. Verifying properties of well-founded linked lists. ACM SIGPLAN Notices, 2006,41(1):115−126. [doi: 10.1145/1111320.1111048]

[10] Lahiri S, Qadeer S. Back to the future: Revisiting precise program verification using SMT solvers. ACM SIGPLAN Notices, 2008, 43(1):171−182. [doi: 10.1145/1328897.1328461]

[11] Cao J, Fu M, Feng X. Practical tactics for verifying C programs in Coq. In: Leroy X, ed. Proc. of the 2015 Conf. on Certified Programs and Proofs. Mumbai: ACM Press, 2015. 97−108. [doi: 10.1145/2676724.2693162]

[12] Gries D, Prins J. A new notion of encapsulation. ACM SIGPLAN Notices, 1985,20(7):131−139. [doi: 10.1145/17919.806834]

[13] Zhao JH, Li XD. Scope logic: An extension to Hoare logic for pointers and recursive data structures. In: Liu ZM, ed. Proc. of the Theoretical Aspects of Computing (CICTAC 2013). Shanghai: Springer-Verlag, 2013. 409−426. [doi: 10.1007/978-3-642-39718-9_24]

[14] Back RJR. A calculus of refinements for program derivations. Acta Informatica, 1988,25(6):593−624. [doi: 10.1007/BF00291051]

[15] Back RJR. Correctness preserving program refinements: Proof theory and applications. MC Tracts, 1980,131:1−118.

[16] Morgan C, Robinson K. Specification statements and refinement. IBM Journal of Research and Development, 1987,31(5):546−555. [doi: 10.1147/rd.315.0546]

[17] Morris JM. A theoretical basis for stepwise refinement and the programming calculus. In: Sintzoff M, ed. Proc. of the Science of Computer programming. Amsterdam: Elsevier, 1987. 287−306. [doi: 10.1016/0167-6423(87)90011-6]

[18] Morgan C. The specification statement. ACM Trans. on Programming Languages and Systems (TOPLAS), 1988,10(3):403−419. [doi: 10.1145/44501.44503]

[19] Gardiner PH, Morgan CC. Data refinement of predicate transformers. Theoretical Computer Science, 1991,87(1):143−162. [doi: 10.1016/0304-3975(91)90029-2]

[20] Morgan C. Auxiliary variables in data refinement. Information Processing Letters, 1988,29(6):293−296. [doi: 10.1016/0020-0190(88)90227-X]

[21] Morris JM. Laws of data refinement. Acta Informatica, 1989,26(4):287−308. [doi: 10.1007/BF00276019]

[22] Morgan C, Gardiner PH. Data refinement by calculation. Acta Informatica, 1990,27(6):481−503. [doi: 10.1007/BF00277386]

[23] Chen W, Udding JT. Towards a calculus of data refinement. In: van de Snepscheut JLA, ed. Proc. of the Mathematics of Program Construction. London: Springer-Verlag, 1989. 197−218. [doi: 10.1007/3-540-51305-1_11]

[24] Leuschel M, Butler M. Automatic refinement checking for B. In: Lau KK, ed. Proc. of the Formal Methods and Software Engineering. Manchester: Springer-Verlag, 2005. 345−359. [doi: 10.1007/11576280_24]

[25] Burdy L, Meynadier JM. Automatic refinement. In: Proc. of the BUGM at FM. 1999. 99. https://link.springer.com/book/10.1007%2F3-540-48118-4

[26] Cohen E, Dahlweid M, Hillebrand M, Leinenbach D, Moskal M, Santen T, Schulte W, Tobies S. VCC: A practical system for verifying concurrent C. In: Berghofer S, ed. Proc. of the Theorem Proving in Higher Order Logics. Munich: Springer-Verlag, 2009. 23−42. [doi: 10.1007/978-3-642-03359-9_2]

[27] Leino KR. Dafny: An automatic program verifier for functional correctness. In: Clarke EM, ed. Proc. of the Logic for Programming, Artificial Intelligence, and Reasoning. Dakar: Springer-Verlag, 2010. 348−370. [doi: 10.1007/978-3-642-17511-4_20]

[28] Condit J, Hackett B, Lahiri SK, Qadeer S. Unifying type checking and property checking for low-level code. In: Shao Z, ed. Proc. of the Principles of Programming Languages. Savannah: ACM Press, 2009. 302−314. [doi: 10.1145/1594834.1480921]

[29] Distefano D, Parkinson JMJ. jStar: Towards practical verification for Java. ACM Sigplan Notices, 2008,43(10):213−226. [doi: 10.1145/1449955.1449782]

[30] Berdine J, Calcagno C, O'hearn PW. Smallfoot: Modular automatic assertion checking with separation logic. In: de Boer FS, ed. Proc. of the Formal Methods for Components and Objects. Amsterdam: Springer-Verlag, 2006. 115−137. [doi: 10.1007/11804192_6]

附录:Schorre-Waite 验证义务的证明过程

本节整体描述 Schorre-Waite 程序上验证义务(如图 8 所示)的证明过程.Schorre-Waite 的一致性证明分解后变成对若干基本块的证明,基本块内的语句主要是对栈(rotate list 实现)的头部进行操作,这些操作较为简单,很多工作或工具可以较为容易地进行证明.这里使用 scope logic 工具进行证明.下面给出大概的证明过程.

下面的证明使用 v'表示 $M(v)$.$M(v)$表示抽象变量 v 对应的具体变量表达式.$NS0$ 表示节点图中所有节点引用的集合.下面给出证明过程中用到的符号的定义.

$$explored=\{x\mid x\in NS0\wedge x\to mk=\text{true}\},$$

$$stack=[t]+STACK(p),$$

$$CHK=\{(x,x\to chk)\mid x\in NS0\},$$

$$INV=IsRotateList(p)\wedge\underset{x\in STACK(p)}{\wedge}(x\to mk),$$

$$STACK(x:P(Node)):List(P(Node))\triangleq x=NULL?[]:x+STACK(x\to chk?x\to r:x\to l),$$

$$IsRotateList(x:P(Node)):bool\triangleq(x=NULL)?\text{true}:IsRotateList(x\to chk?x\to r:x\to l).$$

A.1 C2处验证义务的证明过程(如图9所示)

Pre:
ForAll x in NS0()⇒x→mk=false && x→chk=false∧
root!=null∧root isIn NS0()
*C*1:
t=*root*;
p=*NULL*;
C2:
explored′={}∧stack′=[root]∧p=NULL∧CHK′=CHK′@C1∧INV

Fig.9 Proof obligations of first two lines

图 9 Schorre-Waite 程序前两行的验证义务

使用工具求解 *C*2 处的每个验证义务在 *C*1 处的最弱前置条件.

在 *C*1 处的求解得到的最弱前置条件是

$$explored'=\{\}\wedge[root]+[]=[root]\wedge\text{true}\wedge CHK'=CHK'@C1\wedge IsRotateList(NULL)\wedge\underset{x\in STACK(NULL)}{\wedge}(x\to mk),$$

其中,$CHK'=CHK'@C1$, $IsRotateList(NULL)\wedge\underset{x\in STACK(NULL)}{\wedge}(x\to mk)$ 在 *C*1 处恒成立.

通过前置条件,可以推出 $explored'=\{\}$成立.

A.2 C5和C6处验证义务的证明过程(如图10所示)

使用工具求解 *C*6 处的每个验证义务在 *C*5 处的最弱前置条件.

在 $C5$ 处的求解得到的最弱前置条件是

$$[p]+STACK(p\to r)=tail([t]+STACK(p)) \tag{19}$$

$$p\to r=head(tail(tail([t]+STACK(p)))) \tag{20}$$

$$IsRotateList(p\to r)\wedge \underset{x\in STACK(p\to r)}{\wedge}(x\to mk) \tag{21}$$

其中,$C5$ 处 $p\to chk$ 成立,所以 $STACK(p)=[p]+STACK(p\to r)$成立,所以性质(19)、性质(20)可证.又因为 INV 在 $C5$ 处成立,所以性质(21)成立.

C5: (p,true) isIn CHK′∧stack′=stack′@C4∧explored′=explored′@C4∧CHK′=CHK′@C4∧p=p@C4∧INV
$q=t$; $t=p$; $p=p\to r$; $t=r=q$;
C6: stack′=tail(stack′)@C5∧p=head(tail(tail(stack′)@C5))@C5∧explored′=explored′@C5∧CHK′=CHK′@C5∧INV

Fig.10 Proof obligations of $C5$ and $C6$

图 10 $C5$ 和 $C6$ 处的验证义务

A.3 C7和C8处验证义务的证明过程(如图11所示)

C7: not ((p,true) isIn CHK′)∧stack′=stack′@C4∧explored′=explored′@C4∧CHK′=CHK′@C4∧p=p@C4∧INV
$q=t$; $t=p\to r$; $p\to r=p\to l$; $p\to l=q$; $p\to chk$=true;
C8: stack′=(p→r)@C7+tail(stack′)@C7∧(p@C7,true) isIn CHK′∧explored′=explored′@C7∧INV

Fig.11 Proof obligations of $C7$ and $C8$

图 11 $C7$ 和 $C8$ 处的验证义务

首先,在 $C8$ 处证明 $STACK(p)=STACK(p)@C7$ 成立.因为 $C8$ 处 $p\to chk$ 成立,所以,

$$STACK(p)=[p]+STACK(p\to r) \tag{22}$$

成立.只需证明公式(23)成立.

$$[p]+STACK(p\to r)=STACK(p)@C7 \tag{23}$$

公式(23)在 $C7$ 处的最弱前置条件为

$$[p]+STACK(p\to l)=STACK(p) \tag{24}$$

在 $C7$ 处,$\neg p\to chk$ 成立,所以公式(24)成立.

下面分别证明 $C8$ 处的验证义务.

1) 将验证义务 $stack'=(p\to r)@C7+tail(stack')@C7\wedge \underset{x\in STACK(p)}{\wedge}(x\to mk)$ 中的 $STACK(P)$替换为

$$STACK(P)@C7,$$

求解它们在 $C7$ 处的最弱前置条件,它们的最弱前置条件是

$$[p\to r]+STACK(p)=[p\to r]+tail([t]+STACK(p))\wedge \underset{x\in STACK(p)}{\wedge}(x\to mk) \tag{25}$$

根据 $C7$ 处的性质,公式(25)显然成立;

2) $p=p@C7$ 在 $C8$ 处成立(通过最弱前置条件可证),因为 $p\to chk$ 成立,故$(p@C7,\text{true})\in CHK'$在 $C8$ 处成立;

3) $explored'=explored'@C7$ 通过最弱前置条件可证;

4) 在 $C8$ 处,因为 $p\to chk$ 成立,所以要想证明 $IsRotateList(p)$,只需证明 $IsRotateList(p\to r)$成立.

$IsRotateList(p\to r)$在 $C7$ 处的最弱前置条件是 $IsRotateList(p\to l)$.$IsRotateList(p\to l)$在 $C7$ 处成立.

A.4 C10和C11处验证义务证明过程(如图12所示)

C10: not (head(stack′)=NULL||head(stack′) isIn explored′)∧INV∧
stack′=stack′@C3∧explored′=explored′@C3∧CHK′=CHK′@C3∧p=p@C3
$q=p$; $p=t$; $t=t\to l$; $p\to l=q$; $p-mk$=true; $p\to chk$=false;
C11: p=head(stack′)@C10∧stack′=((head(stack′)@C10)→l)@C10+stack′@C10∧
explored′=explored′@C10+head(stack′)@C10∧(head(stack′)@C10,false) isIn CHK′∧INV

Fig.12 Proof obligations of $C10$ and $C11$

图 12 $C10$ 和 $C11$ 处的验证义务

首先,在 $C11$ 处证明 $STACK(p)=([t]+STACK(p))@C10$ 成立.

因为 $C11$ 处 $\neg p\rightarrow chk$,所以 $STACK(p)=[p]+STACK(p\rightarrow l)$成立.

所以要证明 $STACK(p)=([t]+STACK(p))@C10$,只需证明公式(26)成立.

$$[p]+STACK(p\rightarrow l)=([t]+STACK(p))@C10 \tag{26}$$

公式(26)在 $C10$ 处的最弱前置条件是 true,因此 $STACK(p)=([t]+STACK(p))@C10$ 在 $C11$ 处成立.下面分别证明 $C11$ 处的验证义务.

1) $p=head(stack)@C10$ 在 $C10$ 处的最弱前置条件是 $t=head([t]+STACK(p))$,因此 $p=head(stack)@C10$ 在 $C11$ 处成立;

2) 将 $C11$ 处的验证义务 $stack'=((head(stack')@C10)\rightarrow l)@C10+stack'@C10\wedge\underset{x\in STACK(p)}{\wedge}(x\rightarrow mk)$ 中的 $STACK(P)$替换为$([t]+STACK(p))@C10$,求解它们的最弱前置条件,其最弱前置条件是

$$[t\rightarrow l]+[t]+STACK(p)=[(head([t]+STACK(p)))\rightarrow l]+[t]+STACK(p)\wedge\underset{x\in[t]+STACK(p)}{\wedge}(t\neq x?x\rightarrow mk:\text{true}) \tag{27}$$

根据 $C10$ 处的性质,公式(27)可证;

3) 由 $p=head(stack)@C10$ 和$\neg p\rightarrow chk$ 推出$(head(stack)@C10,\text{false})\in CHK'$在 $C11$ 处成立;

4) 在 $C11$ 处,因为$\neg p\rightarrow chk$,所以要想证明 $IsRotateList(p)$,只需证明 $IsRotateList(p\rightarrow l)$成立.

$IsRotateList(p\rightarrow l)$在 $C10$ 处的最弱前置条件是 $IsRotateList(p).IsRotateList(p)$在 $C11$ 处成立;

5) 在 $C11$ 处,因为 $p\rightarrow mk$ 和 $p=head(stack)@C10$ 成立,所以 $explored'=EXPLORED(NS0-\{p\})+\{p\}$成立.其中,$EXPLORED$ 定义如下:

$$EXPLORED(S:\boldsymbol{Set}(\boldsymbol{P}(\boldsymbol{Node}))):\boldsymbol{Set}(\boldsymbol{P}(\boldsymbol{Node}))\triangleq\{x|x\in S\wedge x\rightarrow mk=\text{true}\}.$$

$EXPLORED(NS0-\{p\})+\{p\}=EXPLORED(NS0)@C10+\{p\}$在 $C10$ 处的最弱前置条件是

$$EXPLORED(NS0-\{t\})+\{t\}=EXPLORED(NS0)+\{t\}.$$

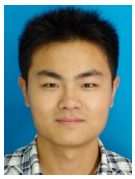
在 $C10$ 处,因为$\neg t\rightarrow mk$ 成立,所以 $EXPLORED(NS0-\{t\})=EXPLORED(NS0)$成立.

因此,$explored'=explored'@C10+head(stack')@C10$ 在 $C11$ 处成立.

综上,Schorre-Waite 的一致性证明分解后变成对若干基本块的证明,它们可以通过求解最弱前置条件,或者稍微作一些变换后求解最弱前置条件来证明.证明过程相对简单.

李彬(1988－),男,河北邯郸人,博士生,主要研究领域为软件工程,程序分析,程序验证.

翟娟(1988－),女,博士,主要研究领域为软件工程,程序分析,程序验证,程序合成.



汤震浩(1989－),男,博士生,主要研究领域为软件工程,程序分析,程序验证.

赵建华(1971－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为形式化方法,软件工程,程序设计语言.