Online Appendix to: Refining Cache Behavior Prediction Using Cache Miss Paths

KARTIK NAGAR and Y. N. SRIKANT, Indian Institute of Science

A. PROOF OF THEOREM 3.3

A function $f : \mathcal{L} \rightarrow \mathcal{L}$ is called distributive if, given $L \subseteq \mathcal{L}$, $f(\bigcup_{P\in L} P) = \bigcup_{P\in L} f(P)$.

LEMMA A.1. *The transfer function f_w is distributive for all basic blocks w.*

PROOF. Since the transfer function $f_w(P)$ (for all cases) operates individually on every $\pi \in P$, $f_w(P) = \bigcup_{\pi\in P} f_w(\{\pi\})$. Given $L \subseteq \mathcal{L}$,

$$\bigcup_{P\in L} f_w(P) = \bigcup_{P\in L} \bigcup_{\pi\in P} f_w(\{\pi\}) \tag{9}$$

$$= \bigcup_{\pi\in\bigcup_{P\in L} P} f_w(\{\pi\}) \tag{10}$$

$$= f_w\left(\bigcup_{P\in L} P\right). \quad \square \tag{11}$$

It is known that in an AI framework, if the individual transfer functions are distributive, then the abstract fixpoint value OUT_w is equal to the join over all paths (JOP) of all abstract values possible at the start of w. Let w_{end} be the unique end basic block (i.e., $\nexists w$, such that $(w_{end}, w) \in E$). Given a walk $\sigma = v_1 v_2 \ldots v_p$, let $f_\sigma = f_{v_1} \circ f_{v_2} \circ \ldots f_{v_p}$ be the cumulative transfer function of σ (in reverse direction). For a basic block w, let Σ_w be the set of all walks in G from w to w_{end}.

LEMMA A.2. *For all basic blocks w, $OUT_w = \bigcup_{\sigma\in\Sigma_w} f_\sigma(\phi)$.*

PROOF. $\bigcup_{\sigma\in\Sigma_w} f_\sigma(\phi)$ is the (backward) JOP over all paths from w to w_{end}, and since the transfer functions are distributive, this will be equal to OUT_w computed using fixpoint-based (backward) analysis. $\square$

LEMMA A.3. *Given a concrete cache miss path $\sigma = v_1 v_2 \ldots v_p v$ of access to m in v, $\alpha^T_{v,m}(\sigma) \in f_{v_1} \circ f_{v_2} \circ \ldots \circ f_{v_p}(\{\{v\}\})$.*

PROOF. Consider the case when $|Acc_a(v_1, m) \cup \bigcup_{i=2}^{p} Acc(v_i, s) \cup Acc_b(v, m)| \geq k$. Also, suppose $|\alpha^T_{v,m}(\sigma)| \leq T$. We will show that for all i, $1 \leq i \leq p$, $\exists\pi \in f_{v_i} \circ f_{v_{i+1}} \circ \ldots \circ f_{v_p}(\{v\})$ such that $\alpha^T_{v,m}(v_i \ldots v_p v) = \pi$. We show this using induction on $p - i$. For $p - i = 0$, that is, for f_{v_p}, only Cases 1 and 2 of the transfer function will apply. If $Acc(v_p, s) = \phi$, then $\alpha^T_{v,m}(v_p v) = \{v\}$, and hence the statement trivially holds. If $Acc(v_p, s) \neq \phi$, then $\alpha^T_{v,m}(v_p v) = \{v_p, v\}$, but then Case 2 applies and v_p will be added to $\pi = \{v\}$.

Now, assume the inductive hypothesis holds for some $p - i$. We want to show the result for $p - (i - 1)$. If $i > 1$, then again only Cases 1 and 2 apply. If $Acc(v_{i-1}, s) = \phi$, then $\alpha^T_{v,m}(v_{i-1} \ldots v_p v) = \alpha^T_{v,m}(v_i \ldots v_p v)$. Also, $f_{v_{i-1}} \circ f_{v_i} \circ \ldots \circ f_{v_p}(\{v\}) = f_{v_i} \circ \ldots \circ f_{v_p}(\{v\})$ (by Case 1). Hence, by the inductive hypothesis, $\exists\pi \in f_{v_{i-1}} \circ f_{v_i} \circ \ldots \circ f_{v_p}(\{v\})$ such that

DOI: http://dx.doi.org/10.1145/3035541

$\alpha^T_{v,m}(v_{i-1}\dots v_p v) = \pi$. If $Acc(v_{i-1}, s) \neq \phi$, then $\alpha^T_{v,m}(v_{i-1}\dots v_p v) = \{v_{i-1}\} \cup \alpha^T_{v,m}(v_i \dots v_p v)$. However, v_{i-1} will also be added π in $f_{v_i} \circ \dots \circ f_{v_p}(\{v\})$ (by Case 2).

Finally, consider the case when $i = 1$. Now, only Cases 3 and 4 apply. If $v_1 = v$, then $\alpha^T_{v,m}(v_1 v_2 \dots v_p v) = \alpha^T_{v,m}(v_2 \dots v_p v)$. Hence, by inductive hypothesis, $\exists \pi \in f_{v_2} \circ \dots \circ f_{v_p}(\{v\})$ such that $\alpha^T_{v,m}(v_2 \dots v_p v) = \pi$. Case 3 applies and since $DB_{v,m}(\alpha^T_{v,m}(\sigma)) \geq k$, $\pi \in f_{v_1}(\{\pi\})$. If $v_1 \neq v$, then, since no suffix of the concrete cache miss path is also a concrete cache miss path, v_1 will be added to π by the transfer function f_{v_1} (Case 4). This completes the proof for the case when $|\alpha(\sigma)| \leq T$ and $v_1 \neq v_{start}$. The proof for the two remaining cases (i.e., $v_1 = v_{start}$ and $|\alpha(\sigma)| = T$) will be similar. □

THEOREM 3.3. *For every concrete cache miss path σ of access r in basic block v, there exists an abstract cache miss path $\pi \in OUT_{v_{start}}$ such that $\pi = \alpha^T_{v,m}(\sigma)$.*

PROOF. Let $\sigma = v_1 \dots v_p v$. Let σ_e be a walk in G from v to w_{end} that does not pass through v. Then $f_{\sigma_e}(\phi) = \phi$. By Lemma A.3, $\alpha^T_{v,m}(\sigma) \in f_{v_1} \circ f_{v_2} \circ \dots \circ f_{v_p}(f_v(\phi)) = f_\sigma(\phi)$. If $v_1 = v_{start}$, then $\sigma\sigma_e$ is a walk from v_{start} to w_{end}, and hence, by Lemma A.2, $f_{\sigma\sigma_e}(\phi) \in OUT_{v_{start}}$.

If $v_1 \neq v_{start}$, then let σ_s be a walk from v_{start} to v_1. Now, either $DB_{v,m}(\alpha^T_{v,m}(\sigma)) \geq k$ or $|\alpha^T_{v,m}(\sigma)| = T$, and hence, for all w in σ_s, $f_w(\{\alpha^T_{v,m}(\sigma)\}) = \{\alpha^T_{v,m}(\sigma)\}$. Hence, $\alpha^T_{v,m}(\sigma) \in f_{\sigma_s\sigma\sigma_e}(\phi)$. Again, by Lemma A.2, this means that $\alpha^T_{v,m}(\sigma) \in OUT_{v_{start}}$. □

B. PROOFS OF LEMMAS AND THEOREMS IN SECTION 4

THEOREM 4.1. *If an access to m in v does not have any abstract cache miss paths, then it is guaranteed to cause a cache hit.*

PROOF. By Theorem 3.3, if m does not have any abstract cache miss paths, then it also does not have any concrete cache miss paths. This implies that it can never cause a cache miss. □

THEOREM 4.2. *If an access to m in v does not have any abstract cache miss paths that are completely inside an enclosing loop L, then m is persistent in loop L.*

PROOF. If the access m has an abstract cache miss path, then this path must contain a basic block that is outside the loop L. This implies that m cannot have a concrete cache miss path completely inside L. Hence, m can cause at most one cache miss for every entry to the loop L from outside the loop. □

LEMMA 4.4. *Given a set of basic blocks $W = \{v_1, \dots, v_n\}$ and basic block v ($v \notin W$), if $\forall v_i, v_j \in W$, there exists a walk in G either from v_i to v_j or v_j to v_i that does not pass through v, and then there exists a walk in G that contains all the basic blocks in W and also does not pass through v.*

PROOF. We use induction on the size of the set W. If the size is 1, then the statement is trivial. Suppose the result holds when the size is k. Let $W = \{v_1, \dots, v_k, v_{k+1}\}$. By inductive hypothesis, assume that there exists a walk σ in G that contains all basic blocks from v_1 to v_k (in increasing order). We know that if $\forall i$, there exists a walk in G either from v_{k+1} to v_i or v_i to v_{k+1} that does not pass through v. Let j be the maximum subscript such that there is a walk from v_j to v_{k+1}. Now consider the subwalk of σ from v_1 to v_j, followed by the walk from v_j to v_{k+1}, followed by the walk from v_{k+1} to v_{j+1}, followed by the subwalk of σ from v_{j+1} to v_k. This is a walk in G that contains all basic blocks from W and does not pass through v. This proves the result. □

LEMMA 4.5. *Miss paths* π_1 *and* π_2 *of two accesses in* v *do not conflict* $\Leftrightarrow \forall w_1 \in \pi_1$, $\forall w_2 \in \pi_2$, *and there exists a walk in G either from* w_1 *to* w_2 *or from* w_2 *to* w_1 *that does not pass through* v.

PROOF. The forward direction is trivial, since we can take the required subwalk from the walk σ that contains all basic blocks of π_1 and π_2. For the reverse direction, we simply take $W = \pi_1 \cup \pi_2$ and apply Lemma 4.4, which implies that there is a walk in G that contains all the basic blocks of π_1 and π_2 and does not pass through v. Note that by definition of miss paths, there always exists a walk between two basic blocks of the same miss path that does not pass through v, and there is a walk in G from every basic block in the miss path to v. This shows that π_1 and π_2 do not conflict with each other. □

LEMMA 4.6. *Given miss paths* π_1 *and* π_2 *of two accesses in* v, π_1 *and* π_2 *do not conflict* $\Leftrightarrow \forall w_1 \forall w_2 \in \pi_1 \cup \pi_2,\ (w_1 \in IN_{w_2} \vee w_2 \in IN_{w_1})$.

PROOF. By Lemma 4.5 and the correctness of the DFA $\mathcal{D}_v$. □

LEMMA 4.7. *Given miss paths* $\pi_1, \ldots, \pi_n$, *of accesses in* v, *there exists a walk in G that contains all the miss paths and contains* v *at the end if and only if there is no pairwise conflict in the set* $\{\pi_1, \ldots, \pi_n\}$.

PROOF. The forward direction is trivial, because if there exists a walk that contains every basic block of all miss paths, then it will contain a walk between every pair of basic blocks that does not pass through v, and hence none of the miss paths will conflict with each other. For the reverse direction, we take $W = \cup_{i=1}^{n} \pi_i$. Since there is no pairwise conflict between the miss paths, by Lemma 4.5, there exists a walk between v_i and v_j that does not pass through v, $\forall v_i, v_j \in W$. By Lemma 4.4, this means that there exists a walk in G that contains all the basic blocks of W and does not pass through v. □

THEOREM 4.8. *Given the MPCG* G_M *of basic block* v, *the size of the maximum clique in* G_M *is an upper bound on the maximum number of cache misses that can occur in* v.

PROOF. Suppose $\{r_1, \ldots, r_m\}$ is a set of accesses in v that can become misses together. Then, there exists a concrete cache miss path σ_i for each r_i such that a walk in G contains all the concrete miss paths σ_i. By Theorem 3.3, for every σ_i, there exists an abstract cache miss path $\pi_i = \alpha^T_{v,r_i}(\sigma_i)$. This implies that there exists a walk in G that contains all the basic blocks of π_i (for all i, $1 \le i \le m$). By Lemma 4.7, this means that there is no pairwise conflict in the set $\{\pi_1, \ldots, \pi_m\}$, and hence these abstract cache miss paths will form a clique in the MPCG G_M of v. □

LEMMA 4.9. *Miss paths* π_1 *of access* r_1, π_2 *of* $r_2, \ldots, \pi_k$ *of* r_k *in* v *can cause* k *misses in* v *in consecutive iterations of* $L \Leftrightarrow$ *there exists a walk from* v *to* v *that contains exactly one instance of* v_h *and contains all the miss paths.*

PROOF. If v is executed in a iteration, it will bring all the cache blocks accessed by $r_1, \ldots, r_k$ to the cache. Hence, for these accesses to miss the cache in the next iteration, the miss paths should all occur before v is executed in the next iteration, which will require a walk from v to v containing all the miss paths and passing through v_h once. On the other hand, if such a walk exists, then an execution along this walk can result in k misses in v in consecutive iterations. □

LEMMA 4.10. *Given miss paths* π_1 *of access* $r_1, \ldots, \pi_k$ *of access* r_k *in* v, *there exists a walk from* v *to* v *containing all the miss paths and exactly one instance of* $v_h \Leftrightarrow \forall v_1, v_2 \in \cup_{i=1}^{k} \pi_i,\ v_1 \rightsquigarrow_v v_2 \vee v_2 \rightsquigarrow_v v_1$ *and* $\forall v_1, v_2 \in \cup_{i=1}^{k} \pi_i \cup \{v\},\ v_1 \rightsquigarrow_{v_h} v_2 \vee v_1 \rightsquigarrow_{v_h} v_2$.

Proof. Let $W = \cup_{i=1}^{k}\pi_i$. Let σ be the walk from v to v containing all miss paths and one instance of v_h. The first part of the forward direction is trivial, since all the basic blocks in W will be present in the walk, and since v only occurs at the endpoints of the walk, there must be a walk between every pair of basic blocks in W that does not pass through v. We partition W into two sets $W_{\rightarrow}$ and $W_{\leftarrow}$, such that $W_{\rightarrow}$ contains all basic blocks of W that occur on σ before v_h, and $W_{\leftarrow}$ contains all basic blocks of W that occur on σ after v_h. Then $v \rightsquigarrow_{v_h} v'$ for all $v' \in W_{\rightarrow}$ and $v' \rightsquigarrow_{v_h} v$ for all $v' \in W_{\leftarrow}$. Also, for all v_1, v_2 in $W_{\rightarrow}$, either $v_1 \rightsquigarrow_{v_h} v_2$ or $v_2 \rightsquigarrow_{v_h} v_1$. Similarly, for all v_1, v_2 in $W_{\leftarrow}$, either $v_1 \rightsquigarrow_{v_h} v_2$ or $v_2 \rightsquigarrow_{v_h} v_1$. Finally, for all $v_1 \in W_{\leftarrow}$, $v_2 \in W_{\rightarrow}$, $v_1 \rightsquigarrow_{v_h} v \rightsquigarrow_{v_h} v_2 \Rightarrow v_1 \rightsquigarrow_{v_h} v_2$. This proves the forward direction.

For the reverse direction, we redefine $W_{\rightarrow}$ and $W_{\leftarrow}$ as follows : $W_{\rightarrow} = \{w \in W | v \rightsquigarrow_{v_h} w\}$ and $W_{\leftarrow} = \{w \in W | w \rightsquigarrow_{v_h} v\}$. Now, $\forall v_1, v_2 \in W_{\rightarrow}$, $v_1 \rightsquigarrow_{v_h} v_2 \vee v_2 \rightsquigarrow_{v_h} v_1$. Assume that $v_1 \rightsquigarrow_{v_h} v_2$. This walk will also not pass through v, because otherwise $v_1 \rightsquigarrow_{v_h} v$, and this would imply a walk between two instances of v that does not contain v_h, which is a contradiction because v_h is the entry block of the innermost loop containing v. Now, since $\forall v_1, v_2 \in W_{\rightarrow}$ $v_1 \rightsquigarrow_{v_h} v_2 \vee v_2 \rightsquigarrow_{v_h} v_1$, by Lemma 4.4, there exists a walk $\sigma_{\rightarrow}$ that contains all basic blocks in $W_{\rightarrow}$ and does not pass through v_h. This walk will also not pass through v. Similarly, there exists a walk $\sigma_{\leftarrow}$ that contains all basic blocks in $W_{\leftarrow}$ and does not pass through v_h and v. Now, the walk from v to the first basic block in $\sigma_{\rightarrow}$, followed by $\sigma_{\rightarrow}$, followed by the walk from the last basic block in $\sigma_{\rightarrow}$ to v_h, followed by the walk from v_h to the first basic block in $\sigma_{\leftarrow}$, followed by the walk $\sigma_{\leftarrow}$ is the required walk between two instances of v that does contains all basic blocks in W and does not contain v_h. □

Lemma 4.11. *Given basic blocks $w_1, \ldots, w_k$ in loop L (or one of its inner loops), every walk containing these basic blocks contains at least $k-1$ instances of $v_h \Leftrightarrow \forall w_i, w_j$, $1 \leq i < j \leq k$, neither $w_i \rightsquigarrow_{v_h} w_j$ nor $w_j \rightsquigarrow_{v_h} w_i$.*

Proof. We prove the forward direction by contradiction. Suppose every walk containing $w_1, \ldots, w_k$ contains at least $k-1$ instances of v_h. Assume, for the sake of contradiction, that $\exists w_i, w_j$ such that $w_i \rightsquigarrow_{v_h} w_j$. Now consider all basic blocks apart from w_j. Clearly, there exists a walk that contains all these $k-1$ basic blocks that contains $k-2$ instances of v_h and ends at w_i (this is because there exists a walk between every w_l and w_m that passes through v_h). Now, appending the walk between w_i and w_j that does not contain v_h gives a walk containing $k-2$ instances of v_h and all the k basic blocks, which is a contradiction.

The reverse direction can also be proved using contradiction. Suppose $\forall w_i, w_j$, $1 \leq i < j \leq k$, neither $w_i \rightsquigarrow_{v_h} w_j$ nor $w_j \rightsquigarrow_{v_h} w_i$. Assume, for contradiction, that there exists a walk that contains all the basic blocks $w_1, \ldots, w_k$ and $k-2$ instances of v_h. The instances of v_h partition this walk into $k-1$ segments that do not contain v_h. Since all k basic blocks $w_1, \ldots, w_k$ are present in these segments, by the pigeon-hole principle, there must exist at least one segment that contains two basic blocks w_i, w_j. However, this would mean a walk between these basic blocks that does not contain v_h, which contradicts our assumption. □

Theorem 4.13. *Given miss paths π_1 of access $r_1, \ldots,$ π_k of access r_k in v, where $\pi_1, \ldots, \pi_k \in V_M^L$, if there exists $W_C \subseteq \cup_{i=1}^{k}\pi_i \cup \{v\}$ such that $\forall w, w' \in W_C$ neither $w \rightsquigarrow_{v_h} w'$ nor $w' \rightsquigarrow_{v_h} w$, then a walk from v to v containing all the basic blocks in W_C, with v only coming at the endpoints, requires at least $|W_C|$ instances of v_h.*

Proof. Let $n = |W_C|$. First, consider the case where $v \notin W_C$. Since $v \notin W_C$, we know that $\forall w \in W_C$, either $v \rightsquigarrow_{v_h} w$ or $w \rightsquigarrow_{v_h} v$. However, if $\exists w, w' \in W_C$ such that $v \rightsquigarrow_{v_h} w$ and $w' \rightsquigarrow_{v_h} v$, then this would imply $w' \rightsquigarrow_{v_h} w$, which is a contradiction. Hence, either

Table VII. Benchmarks, Code Size, Cache Configurations

Benchmark	Source	Code Size (in Bytes)	Cache Configuration (No. of Sets-Block Size-Associativity)
adpcm	Mälardalen	1,112	8-16-2
binarysearch	Mälardalen	272	4-16-2
expint	Mälardalen	432	4-16-2
compress	Mälardalen	3,880	8-32-4
crc	Mälardalen	2,216	4-32-2
ndes	Mälardalen	4,944	16-16-4
qurt	Mälardalen	2,504	4-32-4
ud	Mälardalen	1,944	8-16-4
countnegative	Mälardalen	840	4-16-2
lms	Mälardalen	3,680	8-32-2
qsort-exam	Mälardalen	1,208	8-16-2
select	Mälardalen	2,776	8-32-2
sqrt	Mälardalen	544	4-16-2
basicmath	MiBench	116,592	32-32-4
susan	MiBench	48,192	32-32-4
fmref	StreamIt	48,336	32-32-4
audiobeam	StreamIt	47,272	32-32-4
Task 1	DEBIE-1	78,488	32-32-4
Task 4	DEBIE-1	152,328	128-32-4
Task 5	DEBIE-1	91,032	64-32-4

there is walk from v to all w in W_C or there is a walk from all w in W_C to v, which does not contain v_h. Suppose all walks are only from v to all basic blocks in W_C. Now, by Lemma 4.11, a walk containing all n basic blocks in W_C requires at least $n-1$ instances of v_h. If w_s is the start basic block and w_e is the end basic block of this walk, then a walk from w_e to v will require another instance of v_h. Hence, a walk from v to v containing all n basic blocks will require n instances of v_h. The case when there is a walk from all w in W_C to v without passing through v_h can be proved in a similar manner.

If $v \in W_C$, then by Lemma 4.11, a walk containing all basic blocks in W_C will require $n - 1$ instances of v_h. If such a walk starts with v and ends with some basic block $w_e \in W_C$, then since there is no walk from w_e to v that does not pass through v_h, for such a walk to end, v will require one more instance of v_h. Similarly, if such a walk ends with v but starts with some basic block $w_s \in W_C$, then a walk from v to w_s will require another instance of v_h. □