Horn Clauses

Klaus v. Gleissenthall

VRIJE UNIVERSITEIT AMSTERDAM

Recap

- Verficiation by computing a set of verification conditions that can be checked by an SMT solver.
- The hard part are loop invariants! We needed to write them by hand.
- Now: Finding inductive loop invariants automatically!
  - First: <u>Horn clauses</u>
  - Great way to think about & represent constraints on loop invariants
  - Also known as Constrained Horn Clauses (CHC)
  - Hugely popular now, but fairly recent invention (2012)
  - Then: Solving Horn clauses = find invariants automatically

Horn Clauses: Motivation

- Let's look again at a simple Hoare triple

$$\vdash \{x > 0\}\ \mathrm{y{:=}x;\ z{:=}x{+}y}\ \{\mathrm{z} > 0\}$$

- In order to prove the triple, we can apply the rule of composition:

$$\frac{\vdash \{P\}\ \mathrm{s}_1\ \{Q\} \qquad \vdash \{Q\}\ \mathrm{s}_2\ \{R\}}{\vdash \{P\}\ \mathrm{s}_1;\ \mathrm{s}_2\ \{R\}}$$

- The rule requires us to find a formula $\boldsymbol{q}(x, y, z)$ such that

$$\vdash \{x > 0\}\ \mathrm{y{:=}x}\ \{\boldsymbol{q}(x,y,z)\} \text{ and } \vdash \{\boldsymbol{q}(x,y,z)\}\ \ \mathrm{z{:=}x{+}y}\ \{\mathrm{z} > 0\}$$

hold. Here $\boldsymbol{q}(x, y, z)$ means that formula $\boldsymbol{q}$ only uses variables x, y, and z

Horn Clauses: Motivation

Quiz:

- The rule requires us to find a formula $q(x, y, z)$ such that

$$\vdash \{x > 0\}\ \texttt{y:=x}\ \{q(x,y,z)\} \text{ and } \vdash \{q(x,y,z)\}\ \texttt{z:=x+y}\ \{z > 0\}$$

hold

- What's a solution for $q(x,y,z)$?

- Is there an automated way to compute $q(x,y,z)$ for this example?

Horn Clauses: Motivation

- The rule requires us to find a formula $\boldsymbol{q}(x, y, z)$ such that

$$(1)\ \vdash \{x > 0\}\ \mathrm{y{:=}x}\ \{\boldsymbol{q}(x,y,z)\} \text{ and } (2) \vdash \{\boldsymbol{q}(x,y,z)\}\ \ \mathrm{z{:=}x{+}y}\ \{\mathrm{z} > 0\}$$

hold

- If we think of $\boldsymbol{q}(x,y,z)$ as a relation in first-order logic, the Hoare triple above encodes the following constraints on $\boldsymbol{q}(x, y, z)$:

$$(1)\quad (x > 0 \land y = x) \rightarrow \boldsymbol{q}(x, y, z)$$

$$(2)\quad (\boldsymbol{q}(x, y, z) \land z' = x + y)) \rightarrow z' > 0$$

- Importantly, there exists some solution $\boldsymbol{q}(x, y, z)$ iff the Hoare-triple is valid

Horn Clauses: Motivation

<u>Example:</u>

$$(x > 0 \wedge y = x) \rightarrow \quad \boldsymbol{q}(x, y, z)$$

$$(\ \boldsymbol{q}(x, y, z) \wedge z' = x + y)) \rightarrow \quad z' > 0$$

- Let's check if $\boldsymbol{q}(x, y, z) \triangleq$ x+y>0 is a solution to the clauses above!

Horn Clauses: Motivation

- Next, let's look at proving a Hoare triple for a while loop W $\triangleq$ while $x < n$ do $x := x+1$
- We want to prove Hoare triple $\vdash \{x \leq n\}$ W $\{x = n\}$
- To prove the triple, we can apply the rule for while loops:

$$\frac{\vdash \{I \wedge b\}\ \mathrm{s}\ \{I\}}{\vdash \{I\}\ \text{while } b \text{ do } s\ \{I \wedge \neg b\}}$$

- This rule requires us to find a formula $\boldsymbol{q}(x, n)$ such that

$$\vdash \{\boldsymbol{q}(x, n) \wedge x < n\}\ x := x+1\ \{\boldsymbol{q}(x, n)\} \text{ and } x \leq n \Rightarrow \boldsymbol{q}(x, n)\ (\boldsymbol{q}(x, n) \wedge x \geq n) \Rightarrow x = n$$

Horn Clauses: Motivation

- This rule requires us to find a formula $q(x, n)$ such that

$$\vdash \{q(x, n) \wedge x < n\}\ x := x + 1\ \{q(x, n)\} \quad \text{and} \quad x \leq n \Rightarrow q(x, n) \quad (q(x, n) \wedge x \geq n) \Rightarrow x = n$$

Quiz:

- Where do the last two conditions come from?

- Can we compute $q(x, n)$ automatically?

Horn Clauses: Motivation

- This rule requires us to find a formula $q(x, n)$ such that

$(1) \vdash \{q(x, n) \wedge x < n\}\ x := x+1\ \{q(x, n)\}$ and $(2)\ x \leq n \Rightarrow q(x, n)$ $(3)\ (q(x, n) \wedge x \geq n) \Rightarrow x = n$

- Again, we can think of $q(x, n)$ as a relation in first-order logic;

  then the Hoare triples above encode the following constraints on $q(x, n)$:

(1) $(q(x, n) \wedge x < n \wedge x' = x+1) \to q(x', n)$

(2) $(x \leq n) \to q(x, n)$

(3) $(q(x, n) \wedge x \geq n) \to x = n$

Horn Clauses: Motivation

<u>Example:</u>

(1) $(\boldsymbol{q}(x, n) \wedge x<n \wedge x'=x+1) \rightarrow \boldsymbol{q}(x', n)$

(2) $(x \leq n) \rightarrow \boldsymbol{q}(x, n)$

(3) $(\boldsymbol{q}(x, n) \wedge x \geq n) \rightarrow x=n$

- Let's check if $\boldsymbol{q}(x, n) \triangleq x \leq n$ is a solution to the clauses above!

Horn Clauses: Syntax

A constrained Horn clause C is a formula of the form

$$\boldsymbol{p}(x_1, x_2) \land \boldsymbol{q}(x_1, x_2, x_3) \land \boldsymbol{r}(x_1) \land \ldots \land \varphi \rightarrow H$$

where

- $\boldsymbol{p}$, $\boldsymbol{q}$, $\boldsymbol{r}$ are zero or more unknown relations called <u>queries,</u> where each queries ranges over a vector of variables
- φ is a formula in a first-order theory (background theory) that doesn't contain queries
- H is called the <u>head</u>, and is either a query or $\bot$
- The left-hand side of the implication is called <u>body</u>
- Free variables are implicitly universally quantified

Horn Clauses: Syntax

Quiz:

- Which of these are well-formed Horn clauses?

- $(x<n) \rightarrow \mathbf{q}(x, n)$
- $\mathbf{q}(x, n) \land (x<n) \rightarrow \bot$
- $\mathbf{q}(x, y) \land \mathbf{r}(x) \rightarrow \mathbf{p}(x)$
- $\mathbf{q}(x, y) \rightarrow \mathbf{p}(x+1, y)$
- $\mathbf{q}(x) \land \neg\mathbf{r}(x) \rightarrow \mathbf{p}(x)$
- $\mathbf{q}(x) \lor \mathbf{r}(x) \rightarrow \mathbf{p}(x)$
- $\mathbf{q}(x) \rightarrow x<n$

$$\mathbf{p}(x_1, x_2) \land \mathbf{q}(x_1, x_2, x_3) \land \mathbf{r}(x_1) \land \ldots \land \varphi \rightarrow H$$

- H is either a query or $\bot$

- Can the last two clauses be transformed into well-formed clauses?

Horn Clauses: Syntax

- Let $\mathbb{C}$ be a set of constrained Horn clauses $\mathbb{C} \triangleq \{C_1, \ldots, C_n\}$
- We say that a query ***r*** depends on query ***q*** if there is some clause C in $\mathbb{C}$ such that ***q*** appears in C's body and ***r*** appears in C's head.
- We say that set $\mathbb{C}$ is <u>recursive</u>, if its dependency graph contains a cycle
- <u>Non-recursive</u> clauses are easy to solve (similar to computing weakest preconditions)
- Solving recursive clauses is undecidable & amounts to <u>finding loop invariants</u>

Horn Clauses: Syntax

Quiz:

- Is the following set of Horn clauses recursive?
  - $\boldsymbol{q}(x) \land \boldsymbol{r}(x) \rightarrow \boldsymbol{p}(x)$
  - $\boldsymbol{p}(x) \land (x<n) \rightarrow \bot$
- How about this one?
  - $\boldsymbol{q}(x) \land \boldsymbol{r}(x) \rightarrow \boldsymbol{p}(x)$
  - $\boldsymbol{p}(x) \land (x>0) \rightarrow \boldsymbol{r}(x)$
  - $\boldsymbol{p}(x) \land (x<n) \rightarrow \bot$

Horn Clauses: Semantics

- A solution is a function Σ that maps queries to formulas in the background theory, over the same variables

- We write $\Sigma \models C$ and say that Σ satisfies C, if C is true if we replace all queries by their solution

$$\frac{\Sigma(\mathbf{p}) \wedge \Sigma(\mathbf{q}) \wedge \ldots \wedge \varphi \Rightarrow \Sigma(\mathbf{r})}{\Sigma \models \mathbf{p}(x_1, x_2) \wedge \mathbf{q}(x_1, x_2, x_3) \wedge \ldots \wedge \varphi \to \mathbf{r}(x_1)}$$

- We write $\Sigma \models \mathbb{C}$ and say that Σ satisfies the set of clauses $\{C_1, \ldots, C_n\}$, if it satisfies all individual clauses, i.e., $\Sigma \models C_1$, and $\Sigma \models C_2$, and …, $\Sigma \models C_n$.

- We say that $\mathbb{C}$ is satisfiable, if there exists a solution Σ, s.t. $\Sigma \models \mathbb{C}$

Horn Clauses: Semantics

Quiz:

- Consider the following set $\mathbb{C}$ of Horn clauses

$$(1) \quad x = 1 \rightarrow \mathbf{p}(x)$$

$$(2) \quad p(x) \wedge x' = x + 1 \rightarrow p(x')$$

$$(3) \quad p(x) \rightarrow 0 \leq x$$

- Is $\Sigma \triangleq \{\mathbf{p} \rightarrow x = 1\}$ a solution to $\mathbb{C}$?

- What about $\Sigma \triangleq \{\mathbf{p} \rightarrow x \geq 1\}$?

From Programs to Horn Clauses

- Idea: we're going to use weakest preconditions to translate programs to clauses
- But now, the weakest precondition is always an unknown predicate, that is a ***query***
- As side-condition, we're going to generate Horn constraints
- Let's start with assignments:

$$\mathrm{wp}(x := e, \mathbf{p}(x_1, \ldots, x_n)) \triangleq \mathbf{p}(x_1, \ldots, x_n)[e/x]$$

- We substitute e for x in the query and produce no extra Horn constraints

From Programs to Horn Clauses

$$\mathrm{wp}(\,x := e, \boldsymbol{p}(x_1, \ldots, x_n)) \triangleq \boldsymbol{p}(x_1, \ldots, x_n)[\mathrm{e}/\mathrm{x}]$$

Quiz:

- What's $\mathrm{wp}(\,x := x+1,\ \boldsymbol{p}(x, y))$?

From Programs to Horn Clauses

- For sequential composition $s_1; s_2$, we get, as before:

$$\mathrm{wp}(s_1; s_2, \boldsymbol{p}(x_1, \ldots, x_n)) \triangleq \mathrm{wp}(s_1, \mathrm{wp}(s_2, \boldsymbol{p}(x_1, \ldots, x_n)))$$

- For an if-statement if b then s_1 else s_2, let $\boldsymbol{q}(x_1, \ldots, x_n)$ be a <u>fresh</u> query:

$$\mathrm{wp}(\text{if b then } s_1 \text{ else } s_2, \boldsymbol{p}(x_1, \ldots, x_n)) \triangleq \boldsymbol{q}(x_1, \ldots, x_n)$$

- And we add the following Horn constraints on $\boldsymbol{q}(x_1, \ldots, x_n)$:

$$\boldsymbol{q}(x_1, \ldots, x_n) \wedge \mathrm{b} \rightarrow \mathrm{wp}(s_1, \boldsymbol{p}(x_1, \ldots, x_n)) \qquad \boldsymbol{q}(x_1, \ldots, x_n) \wedge \neg \mathrm{b} \rightarrow \mathrm{wp}(s_2, \boldsymbol{p}(x_1, \ldots, x_n))$$

Quiz:

- What does the new query $\boldsymbol{q}(x_1, \ldots, x_n)$ represent?

From Programs to Horn Clauses

Quiz:

- Consider the statement s:

$$x := y + 1;\ \text{if } x > 0 \text{ then } z := 1 \text{ else } z := -1$$

- What is wp$(s, \boldsymbol{p}(x, y, z))$?

From Programs to Horn Clauses

- Last, we have while loops: while b do s
- Now, our while loops are <u>no longer</u> annotated with invariants!
- Let $\mathbf{q}(x_1, \ldots, x_n)$ be a <u>fresh</u> query, then:

$$\mathrm{wp}(\text{while } b \text{ do } s, \mathbf{p}(x_1, \ldots, x_n)) \triangleq \mathbf{q}(x_1, \ldots, x_n)$$

Quiz:

- Do we need to add additional Horn constraints?

From Programs to Horn Clauses

- Last, we have while loops: while b do s
- Now, our while loops are <u>no longer</u> annotated with invariants!
- Let $\mathbf{q}(x_1, \ldots, x_n)$ be a <u>fresh</u> query, then:

$$\mathrm{wp}(\text{while } b \text{ do } s, \mathbf{p}(x_1, \ldots, x_n)) \triangleq \mathbf{q}(x_1, \ldots, x_n)$$

- We need to add the following constraints on $\mathbf{q}(x_1, \ldots, x_n)$

$$\mathbf{q}(x_1, \ldots, x_n) \wedge b \rightarrow \quad \mathrm{wp}(s, \mathbf{q}(x_1, \ldots, x_n)) \quad \mathbf{q}(x_1, \ldots, x_n)$$

$$\wedge \neg b \rightarrow \quad \mathbf{p}(x_1, \ldots, x_n)$$

Quiz:

- What does the new query $\mathbf{q}(x_1, \ldots, x_n)$ represent?

From Programs to Horn Clauses

Quiz:

- Let's look again at while loop W $\triangleq$ while $x<n$ do $x := x+1$
- We want to prove Hoare triple $\vdash \{x \leq n\}$ W $\{x = n\}$

- What is wp$(W, \boldsymbol{p}(x, n))$?
- Which clauses do we have to add to prove the triple?
- Do the Horn-clauses match our syntax restriction?

$$\mathrm{wp}(\text{while } b \text{ do } s,\ \boldsymbol{p}(x_1, \ldots, x_n)) \triangleq \boldsymbol{q}(x_1, \ldots, x_n)$$

$$\boldsymbol{q}(x_1, \ldots, x_n) \land b \rightarrow \qquad \mathrm{wp}(\mathrm{s},\ \boldsymbol{q}(x_1, \ldots, x_n))$$

$$\boldsymbol{q}(x_1, \ldots, x_n) \land \neg b \rightarrow \qquad \boldsymbol{p}(x_1, \ldots, x_n)$$

Normalizing Horn Clauses

- The clauses produced by our procedure, <u>don't fit</u> our syntactic restrictions, yet
- Queries may contain arbitrary expressions rather than variables, only
- Idea: normalize clauses by introducing definitions via <u>fresh variables</u>.
- That is, we transform a clause

$$\boldsymbol{p}(e_1, e_2) \wedge \ldots \wedge \varphi \rightarrow \boldsymbol{r}(e_3)$$

- into

$$\boldsymbol{p}(x_1, x_2) \wedge x_1 = e_1 \wedge x_2 = e_2 \wedge x_3 = e_3 \wedge \ldots \wedge \varphi \rightarrow \boldsymbol{r}(x_3)$$

- Where $x_1, \ldots, x_3$ are fresh variables
- We will now freely use this extended syntax, since we know it can be normalized

Verifying Hoare Triples

- For a Hoare triple $\{P\}$ s $\{Q\}$, let $\boldsymbol{p}(x_1, \ldots, x_n)$ be a query over all variables in s
  - Let wp$(s, \boldsymbol{p}(x_1, \ldots, x_n)) = \boldsymbol{q}(e_1, \ldots, e_n)$
  - Let $\mathbb{C}$ be the set of Horn clauses produced by wp
- The the set $\mathbb{C} \cup \{P \rightarrow \boldsymbol{q}(e_1, \ldots, e_n), \boldsymbol{p}(x_1, \ldots, x_n) \rightarrow Q\}$ is satisfiable if an only if

  $\vdash \{P\}$ s $\{Q\}$, i.e., the Hoare triple is valid

Quiz:
  - How could we prove this?

- Note: Our programs are no longer annotated with loop invariants.
- Determining whether a set of Horn clauses is satisfiable, requires finding loop invariants

Solving Horn clauses

- Next, we will look at <u>solving</u> Horn clauses
- That is, for a set $\mathbb{C}$, we want to compute a solution Σ, s.t. $\Sigma \models \mathbb{C}$
- <u>Undecidable</u>, as it entails finding loop invariants
- We will look at a <u>semi-automated</u> method that works well in practice

Strongest Postconditions

- We first define an operator post, that computes the strongest postcondition of a formula φ with respect to a set of variables $x_1, \ldots, x_n$

- Let $y_1, \ldots, y_k$ be the variables in φ that are not in $x_1, \ldots, x_n$

$$\text{post}(\varphi, x_1, \ldots, x_n) \triangleq \exists y_1, \ldots, y_k.\ \varphi$$

- You can think of post as the projection of formula φ onto variables $x_1, \ldots, x_n$

- The following property holds for all φ_1, φ_2 and $x_1, \ldots, x_n$:

  - $\text{post}(\varphi_1 \vee \varphi_2, x_1, \ldots, x_n) = \text{post}(\varphi_1, x_1, \ldots, x_n) \vee \text{post}(\varphi_2, x_1, \ldots, x_n)$

Strongest Postconditions

Quiz:

- What is post($y=x+1 \wedge x \geq 0$, y)?

- What is post($y=x+1 \wedge (x=0 \vee x=1)$, y)?

Computing a Solution

- We can use post to compute a solution for a set of Horn clauses as follows
- Initially, we start with a solution Σ that maps every ***query*** to ⊥
- Pick any clause whose head is a query, that is, a clause of the form

$$\boldsymbol{p}(y_1, y_2) \wedge \boldsymbol{q}(y_1, y_2, y_3) \wedge \ldots \wedge \varphi \rightarrow \boldsymbol{r}(x_1, x_2, x_3)$$

- and compute

$$\mathrm{p} \triangleq (\mathrm{post}(\Sigma(\boldsymbol{p}) \wedge \Sigma(\boldsymbol{q}) \wedge \ldots \wedge \varphi, x_1 x_2 x_3))$$

- Then, if $\not\models \mathrm{p} \rightarrow \Sigma(\boldsymbol{r})$, set $\Sigma(\boldsymbol{r}) := (\Sigma(\boldsymbol{r}) \vee \mathrm{p})$

Computing a Solution

Example:

- Let's consider the following set of clauses

(1) $x=0 \rightarrow q(x)$

(2) $(q(y) \wedge y<6 \wedge x=y+1) \rightarrow q(x)$

- We start off with solution $\Sigma \triangleq \{q \rightarrow \bot\}$
- Let's pick the first clause $x=0 \rightarrow q(x)$
- What's post$(x=0, x)$?
- Does post$(x=0, x) \Rightarrow \bot$ hold?
- Our new solution is $\Sigma \triangleq \{q \rightarrow x=0\}$

Computing a Solution

Example:

(1) $x=0 \rightarrow q(x)$

(2) $(q(y) \wedge y<6 \wedge x=y+1) \rightarrow q(x)$

- Next, let's pick clause 2
- What is post($y=0 \wedge y<6 \wedge x=y+1$, x)?
- Does post($y=0 \wedge y<6 \wedge x=y+1$, x) $\Rightarrow$ $x=0$ hold?
- Our new solution is $\Sigma \triangleq \{ q \rightarrow (x=0 \vee x=1)\}$

Computing a Solution

Example:

(1) $x=0 \rightarrow \;\; \boldsymbol{q}(x)$

(2) $(\boldsymbol{q}(y) \wedge y<6 \wedge x=y+1) \rightarrow \;\; \boldsymbol{q}(x)$

- We picked clause (2) a few more times
- Our solution now looks like this $\Sigma \triangleq \{\boldsymbol{q} \rightarrow (x=0 \vee x=1 \vee x=2 \vee x=3 \vee x=4 \vee x=5 \vee x=6)\}$
- What is post($\Sigma(\boldsymbol{q})(y) \wedge y<6 \wedge x=y+1 , x$)?
- Does post($\Sigma(\boldsymbol{q})(y) \wedge y<6 \wedge x=y+1 , x$) $\Rightarrow \;\; \Sigma(\boldsymbol{q})$ hold?
- We're done and our solution no longer changes!
- We've reached a fixed-point

Computing a Solution

Quiz:

- Let's change the clauses by renaming some variables:

$$(1) \quad x=0 \rightarrow \mathbf{q}(x)$$

$$(2) \quad (\mathbf{q}(x) \wedge x<6 \wedge x'=x+1) \rightarrow \mathbf{q}(x')$$

- Our solution is $\Sigma \triangleq \{\mathbf{q} \rightarrow x=0\}$
- What is post$(x=0 \wedge x<6 \wedge x'=x+1\ ,\ x')$?
- What do we add to the solution?

Computing a Solution

Quiz:

- In our algorithm, why do we need to check that $\not\models$ post(φ, x_1, x_2, x_3) $\rightarrow$ $\Sigma(\boldsymbol{r})$?
- What happens if we remove this check?

Tip:

- We can think of our algorithm as computing the set of reachable states for $\boldsymbol{r}$
- The check is called subsumption and ensures that we stop once all states are explored

Computing a Solution

Quiz:

- What happens, if we apply our algorithm to the following clauses?

$$(1) \quad x = 1 \wedge n \geq 1 \rightarrow q(x, n)$$

$$(2) \quad (q(y, n) \wedge x = y + 1 \wedge y < n) \rightarrow q(x, n)$$

- Try out!

Computing a Solution

Quiz:

- What happens, if we apply our algorithm to the following clauses?

(1) $x = 1 \wedge n \geq 1 \rightarrow \mathbf{q}(x, n)$

(2) $(\mathbf{q}(y, n) \wedge x = y + 1 \wedge y < n) \rightarrow \mathbf{q}(x, n)$

- If there are infinitely many states for $\mathbf{q}$, the algorithm won't terminate
- This is usually the case for loops: Loops are often unbounded!
- Our algorithm, as presented, is thus no good for computing loop invariants
- We'll solve this problem in the next lecture using abstraction

Horn Clauses

- Reading:
  - Original paper: https://www7.in.tum.de/~popeea/research/hsf.pldi12.pdf
  - Lecture notes: https://github.com/barghouthi/cs704/blob/master/notes/hornClauses.pdf
  - Survey: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/nbjorner-yurifest.pdf