

Last Time

- ◆ **Abstract interpretation**
 - > Dual of dataflow analysis
 - > Dataflow view emphasizes implementation issues
 - > Abstract interpretation view emphasizes correctness via connection with actual execution
- ◆ **We've seen the Galois connection between concrete and abstract domains**

Today

- ◆ **Show that that abstract fixpoint approximates the concrete fixpoint**
- ◆ **This, and the constraints it imposes on transfer functions, is the main result from this course**

Fixed Point Approximation

- ◆ **We want to show that the abstract fixed point approximates the concrete fixed point**

$$\bigsqcup_{i=0}^{\infty} F_c^i(\perp_c) \quad \bigsqcup_{i=0}^{\infty} F_a^i(\perp_a)$$

$$\alpha \left(\bigsqcup_{i=0}^{\infty} F_c^i(\perp_c) \right) \sqsubseteq_a \bigsqcup_{i=0}^{\infty} F_a^i(\perp_a)$$

- ◆ **Want to show:**

$$\alpha \left(\bigsqcup_{i=0}^{\infty} F_c^i(\perp_c) \right) \sqsubseteq_a \bigsqcup_{i=0}^{\infty} F_a^i(\perp_a) \quad (1)$$

$$\bigsqcup_{i=0}^{\infty} F_c^i(\perp_c) \sqsubseteq_c \gamma \left(\bigsqcup_{i=0}^{\infty} F_a^i(\perp_a) \right) \quad (2)$$

- ◆ **These conditions are global: they talk about the fixed point computation**
- ◆ **We want some local conditions on F_a and F_c**

Cousot and Cousot 77

- ◆ **Cousot and Cousot show that the following conditions are sufficient for proving (1) and (2):**

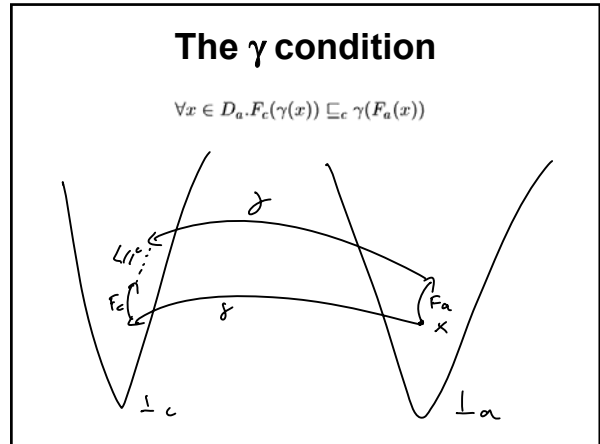
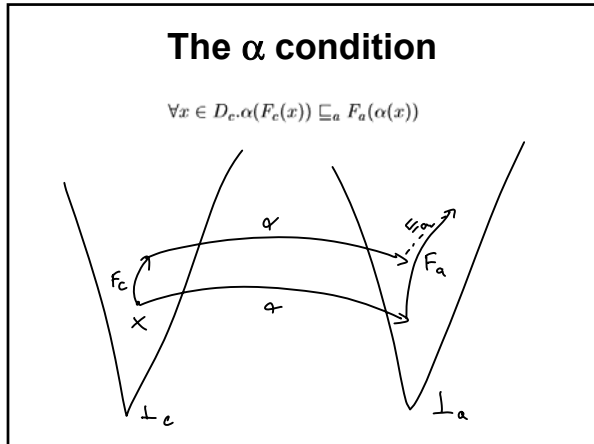
$$\alpha \text{ and } \gamma \text{ are continuous} \quad (3)$$

$$\forall x \in D_c. \alpha(F_c(x)) \sqsubseteq_a F_a(\alpha(x)) \quad (4)$$

$$\forall x \in D_a. F_c(\gamma(x)) \sqsubseteq_c \gamma(F_a(x)) \quad (5)$$

Continuity

- ◆ **Let X be a complete lattice**
- ◆ **A function $f: X \rightarrow X$ is continuous if, for each subset Y of X , we have $\sqcup f(Y) = f(\sqcup Y)$**



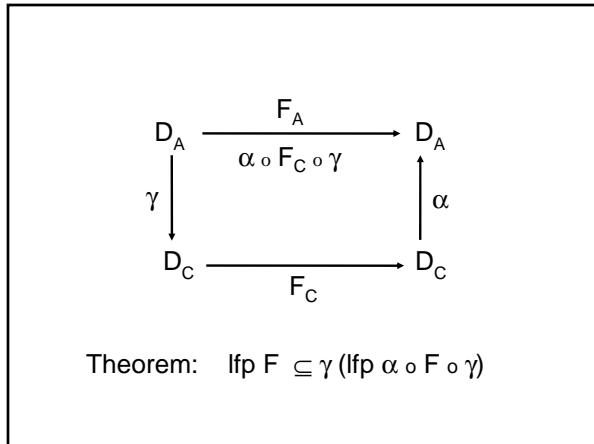
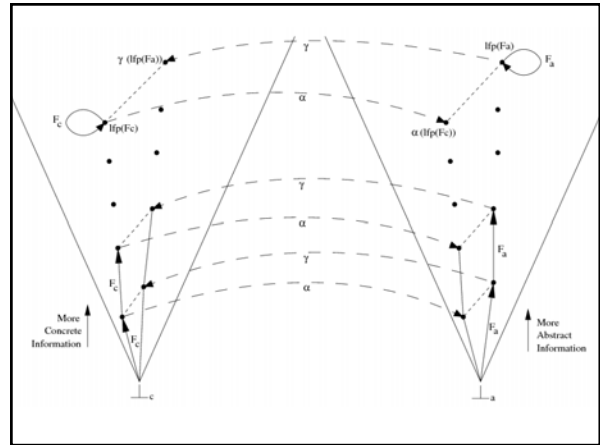
Link between local and global

- $\forall x \in D_c. \alpha(F_c(x)) \sqsubseteq_a F_a(\alpha(x))$ (4)

is local version of

$$\alpha \left(\bigsqcup_{i=0}^{\infty} F_c^i(\perp_c) \right) \sqsubseteq_a \bigsqcup_{i=0}^{\infty} F_a^i(\perp_a)$$
 (1)
- Indeed, using (4) we can show by induction that

$$\forall i \geq 0. \alpha(F_c^i(\perp_c)) \sqsubseteq_a F_a^i(\perp_a)$$



Transfer Functions

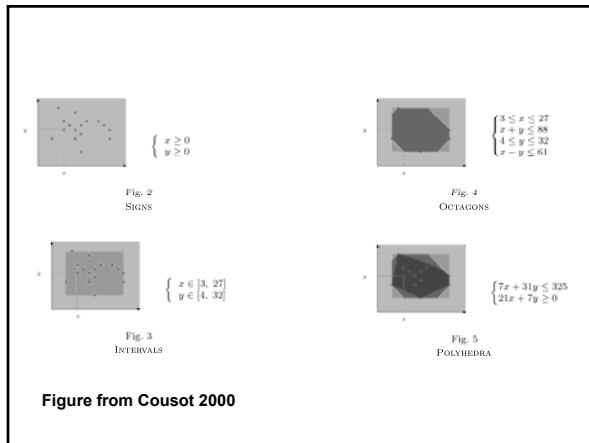
- Basic requirement for local transfer function F_A :**
 - $\alpha \circ F_C \circ \gamma \sqsubseteq_a F_A$
- Can we just use this requirement directly to derive the transfer function?**
 - I.e., concretize the abstract values, apply concrete transfer function, then abstract?
 - Work out example for interval domain addition: $i_1 + i_2 = i_3$
 - What's a more efficient way to add intervals?
- Ok, so where do these transfer functions come from in general?**
 - Thinking hard, usually
 - But at least we know the correctness requirement...

Partitioning

- ◆ We have assumed that program states are partitioned by program points
- ◆ Other partitionings are possible
- ◆ E.g.
 - Partitioning WRT function calls gives context sensitivity
 - Partitioning WRT execution paths gives path sensitivity
 - Failing to partition by program points gives flow insensitivity
 - Etc.

Domains

- ◆ We already looked at domains such as
 - Constants
 - Intervals
 - Bitwise
- ◆ These are all non-relational domains
 - Treat variables in isolation
 - These are fine for supporting optimizations
 - However: Surprisingly weak for computing stronger program properties such as safety invariants
- ◆ Relational domains are much more powerful
 - Unsurprisingly, these track relations among variables



Relational Numerical Domains

- ◆ Transfer functions are usually expensive
 - Polyhedra: Exponential
 - Octagons: Cubic
 - C.f. intervals: Linear
 - How to deal with this in practice?
- ◆ Are unsound in presence of overflow / underflow
 - When does $x > y$ imply that $(x+1) > y$?
 - How to deal with this in practice?
- ◆ Good transfer functions are available: APRON library
 - C and OCaml interfaces supported
 - FP and integer types supported

Decision Tree Abstract Domain

- ◆ Non-numerical relational domain developed for Astree analyzer
 - Astree verified the absence of runtime errors in Airbus A340 fly by wire software
- ◆ Abstract value relates Booleans with a numerical abstract domain
 - Doesn't matter which
- ◆ E.g.
 - $b \rightarrow a=[1..20], \sim b \rightarrow a=[0..19]$
- ◆ Goal: Add some path sensitivity
 - Increases analysis precision
 - Exponential cost

Summary

- ◆ Main result is that the abstract FP approximates the concrete FP
 - And the conditions this imposes on transfer functions
- ◆ Lots of other topics exist...
 - Partitioning the state
 - Widening
 - Domain choice
- ◆ Thurs we'll look at
 - More domains
 - Combining domains