Análisis de Flujo

CI4722 – Lenguajes de Programación III

Ernesto Hernández-Novich
<emhn@usb.ve>

Universidad "Simón Bolívar"

UNIVERSIDAD SIMÓN BOLÍVAR

Análisis de Flujo

- Conjunto de técnicas que permiten derivar información acerca del flujo de los datos sobre los caminos de ejecución.
- Cada una de las optimizaciones globales que discutimos informalmente en la clase pasada resultan de ese análisis.

UNIVERSIDAD SIMÓN BOLÍVAR

Modelo Abstracto de Análisis de Flujo

- Ejecución de un Programa – transformaciones sobre el estado
  - Estado – valores de todas las variables (globales o locales).
  - Transformación – efecto de una instrucción sobre el estado previo (*input state*) para producir el estado posterior (*output state*).
  - Vínculo entre dos **puntos del programa**.

UNIVERSIDAD SIMÓN BOLÍVAR

Modelo Abstracto de Análisis de Flujo

- Ejecución de un Programa – transformaciones sobre el estado
  - Estado – valores de todas las variables (globales o locales).
  - Transformación – efecto de una instrucción sobre el estado previo (*input state*) para producir el estado posterior (*output state*).
  - Vínculo entre dos **puntos del programa**.
- Análisis de Flujo de Datos
  - Considerar **todas** las secuencias **posibles** entre puntos del programa.
  - Extraer la información necesaria para el problema particular a partir de los *conjuntos* de estados **posibles**.
- Análisis Local (*Intraprocedural*) – considera caminos dentro de una unidad de ejecución (programa principal o procedimiento).
- Análisis Global (*Interprocedural*) – estudia caminos que atraviesan varias unidades de ejecución (efecto de llamadas).

UNIVERSIDAD SIMÓN BOLÍVAR

Grafo de Flujo como director

- Bloque Básico – flujo trivial
  - Secuencia lineal de transformaciones.
  - Punto de programa posterior a una instrucción es precisamente el punto previo a la siguiente instrucción.

Grafo de Flujo como director

- Bloque Básico – flujo trivial
  - Secuencia lineal de transformaciones.
  - Punto de programa posterior a una instrucción es precisamente el punto previo a la siguiente instrucción.
- Arista entre B_1 y B_2 – el punto posterior a la *última* instrucción de B_1 **podría** ser el punto previo a la *primera* instrucción de B_2.

UNIVERSIDAD SIMÓN BOLÍVAR

Grafo de Flujo como director

- Bloque Básico – flujo trivial
  - Secuencia lineal de transformaciones.
  - Punto de programa posterior a una instrucción es precisamente el punto previo a la siguiente instrucción.
- Arista entre B_1 y B_2 – el punto posterior a la *última* instrucción de B_1 **podría** ser el punto previo a la *primera* instrucción de B_2.
- **Camino de Ejecución** (*Execution Path*) – o simplemente **Camino**.
  - Secuencia de puntos entre el inicial p_1 y el final p_n.
  - $\forall i : 1 \leq i < n$ el vínculo entre p_i y p_{i+1} es tal que
    - p_i es previo y p_{i+1} posterior a una instrucción.
    - p_i es último de un bloque y p_{i+1} inicial de un sucesor.

Infinitos caminos de ejecución posibles
sin cota superior finita para sus longitudes.

UNIVERSIDAD SIMÓN BOLÍVAR

Gestión de la complejidad

- La técnica se basa en usar un conjunto **finito** de hechos para resumir todos los posibles estados que ocurren en un punto del programa.

Gestión de la complejidad

- La técnica se basa en usar un conjunto **finito** de hechos para resumir todos los posibles estados que ocurren en un punto del programa.
- ¿Cuál información resumir? – Depende del problema de análisis.

UNIVERSIDAD SIMÓN BOLÍVAR

Gestión de la complejidad

- La técnica se basa en usar un conjunto **finito** de hechos para resumir todos los posibles estados que ocurren en un punto del programa.
- ¿Cuál información resumir? – Depende del problema de análisis.
- ¿Podemos resumir *todo*?
  - En general, **no**.
  - Para casos particulares, **a veces no** – los conjuntos no siempre pueden representar perfectamente el estado.

UNIVERSIDAD SIMÓN BOLÍVAR

Gestión de la complejidad

- La técnica se basa en usar un conjunto **finito** de hechos para resumir todos los posibles estados que ocurren en un punto del programa.
- ¿Cuál información resumir? – Depende del problema de análisis.
- ¿Podemos resumir *todo*?
  - En general, **no**.
  - Para casos particulares, **a veces no** – los conjuntos no siempre pueden representar perfectamente el estado.
- Observaciones clave para mejorar nuestras oportunidades.
  - No distinguir entre varios caminos con mismo origen y destino.
  - Representar sólo partes del estado relevantes al análisis.

UNIVERSIDAD SIMÓN BOLÍVAR

Esquema de Análisis de Flujo de Datos

El modelo de datos

- Asociar un **Valor de Flujo** a cada punto del programa.
  - Representación abstracta particular de los estados posibles allí.
  - El conjunto de valores posibles es el **Dominio** de análisis.
- $IN[s]$ y $OUT[s]$ – valores *antes* y *después* de la instrucción s

UNIVERSIDAD SIMÓN BOLÍVAR

¿Cuál es el problema?

Objetivo del análisis

- Problema de Flujo de Datos
  - Dados $IN[s]$ y $OUT[s]$ para *todas* las instrucciones.
  - Encontrar una solución para un *conjunto de restricciones* de manera que se cumplan para todos los $IN[s]$ y $OUT[s]$.
- Dos tipos de restricciones
  - Basadas en la semántica de las instrucciones.
  - Basadas en el flujo de control.

UNIVERSIDAD SIMÓN BOLÍVAR

Restricciones basadas en la semántica

¿Qué efecto tiene la instrucción sobre el estado?

- Denominadas **Funciones de Transferencia**.
- Establecen una relación f_s entre $IN[s]$ y $OUT[s]$
- Propagación hacia adelante (*Forward Propagation*)

$$OUT[s] = f_s(IN[s])$$

- Propagación hacia atrás (*Backward Propagation*)

$$IN[s] = f_s(OUT[s])$$

UNIVERSIDAD SIMÓN BOLÍVAR

Restricciones basadas en el flujo de control

¿Qué efecto tiene el flujo sobre el estado?

- Dentro de un Bloque Básico, el flujo de control no influye

$$IN[s_{i+1}] = OUT[s_I]$$

- Saltos entre Bloques Básicos
  - Restricciones amplían alcance para incluir múltiples orígenes de flujo.
  - $IN[s]$ para la primera instrucción de un bloque depende de uno o más $OUT[s]$ de la(s) última(s) instrucciones de sus predecesores – unión, intersección, diferencia, etc.

UNIVERSIDAD SIMÓN BOLÍVAR

Esquema sobre Bloques Básicos

- El flujo dentro de un Bloque Básico es lineal y continuo – ¡podemos pensar que son una "gran instrucción"!
- Esquema de Análisis de Flujo entre Bloques Básicos
  - $IN[B]$ – valores al entrar al bloque B
  - $OUT[B]$ – valores al salir del bloque B

Necesitamos calcular $IN[B]$ y $OUT[B]$
a partir de $IN[s]$ y $OUT[s], \forall s \in B$

Gran instrucción de pequeñas instrucciones

- Sean $s_1, \ldots, s_n$ las instrucciones de B – en ese orden.

Gran instrucción de pequeñas instrucciones

- Sean $s_1, \ldots, s_n$ las instrucciones de B – en ese orden.
- $IN[B] = IN[s_1]$ – por ser la primera instrucción de B.

Gran instrucción de pequeñas instrucciones

- Sean $s_1, \ldots, s_n$ las instrucciones de B – en ese orden.
- $IN[B] = IN[s_1]$ – por ser la primera instrucción de B.
- $OUT[B] = OUT[s_n]$ – por ser la última instrucción de B.

UNIVERSIDAD SIMÓN BOLÍVAR

Gran instrucción de pequeñas instrucciones

- Sean $s_1, \ldots, s_n$ las instrucciones de B – en ese orden.
- $IN[B] = IN[s_1]$ – por ser la primera instrucción de B.
- $OUT[B] = OUT[s_n]$ – por ser la última instrucción de B.
- Función de Transferencia del Bloque – ¡composición!

$$f_B = f_{s_n} \circ \ldots \circ f_{s_2} \circ f_{s_1}$$

de manera que es natural concluir

$$OUT[B] = f_B(IN[B])$$

UNIVERSIDAD SIMÓN BOLÍVAR

Saliendo del Bloque

- Expresar valores de flujo entre bloques usando $IN[B]$ y $OUT[B]$

Saliendo del Bloque

- Expresar valores de flujo entre bloques usando $IN[B]$ y $OUT[B]$
- "Constantes posiblemente asignadas a una variable" – problema de propagación hacia adelante

$$IN[B] = \bigcup_{P \text{ es predecesor de } B} OUT[P]$$

Saliendo del Bloque

- Expresar valores de flujo entre bloques usando $IN[B]$ y $OUT[B]$
- "Constantes posiblemente asignadas a una variable" – problema de propagación hacia adelante

$$IN[B] = \bigcup_{P \text{ es predecesor de } B} OUT[P]$$

- "Variables vivas" – problema de propagación hacia atrás

$$OUT[B] = \bigcup_{S \text{ es sucesor de } B} IN[S]$$

UNIVERSIDAD SIMÓN BOLÍVAR

Resolución de las Ecuaciones

One does not simply solve data flow equations

- Usualmente tienen múltiples soluciones.
- Se quiere la más "precisa" – aquella tal que:
  - Satisface las restricciones de flujo de control.
  - Satisface las funciones de transferencia.
  - En caso de duda, ser conservador – no cambiar el programa.

UNIVERSIDAD SIMÓN BOLÍVAR

Esquema – Definiciones Vigentes

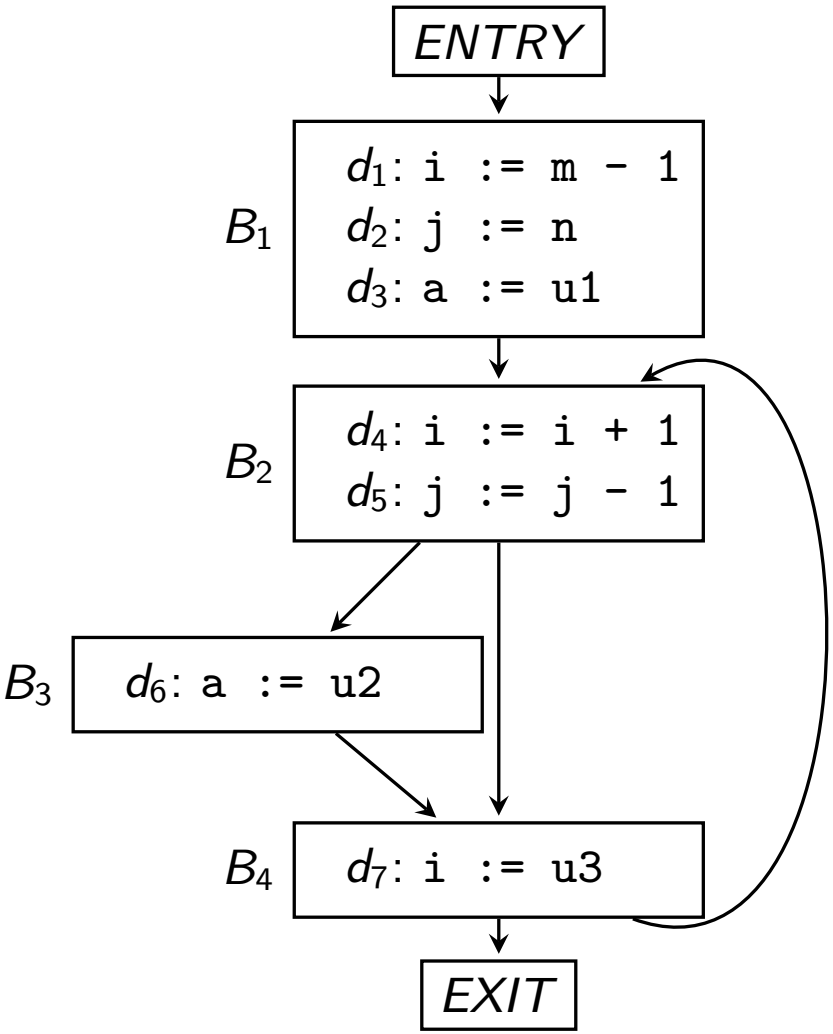
¿Cuál es la definición más reciente para un nombre?

- *Reaching Definitions* – definiciones vigentes en un punto.
- Una definición d está vigente (*reaches*) en el punto p
  - Si hay un camino entre el punto siguiente a d y p.
  - Si d no es redefinida (*killed*) en el camino.
- Una definición d para x es una instrucción que *quizás* asigne valor a x
  - Aliasing no nos permite estar completamente seguros.
  - Somos conservadores en este caso para favorecer *kills* – si no estamos seguros, suponemos que si ocurre un *kill*.

UNIVERSIDAD SIMÓN BOLÍVAR

Reaching Definitions

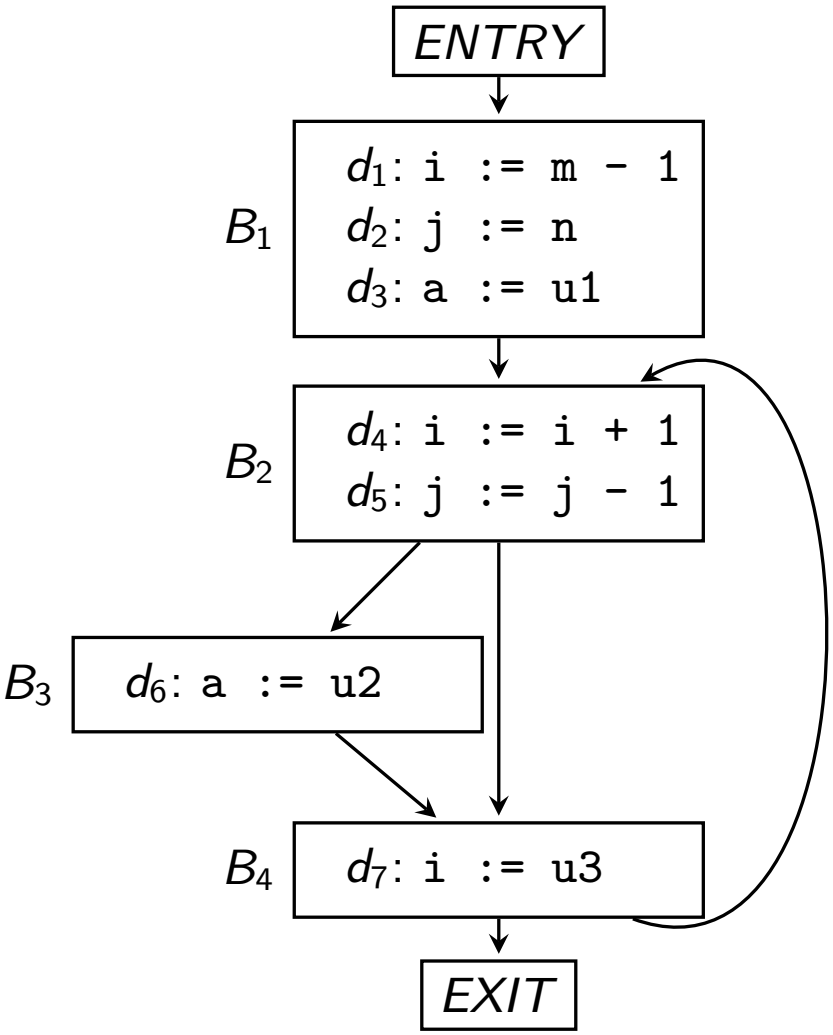
¿Qué queremos determinar?



UNIVERSIDAD SIMÓN BOLÍVAR

Reaching Definitions

¿Qué queremos determinar?

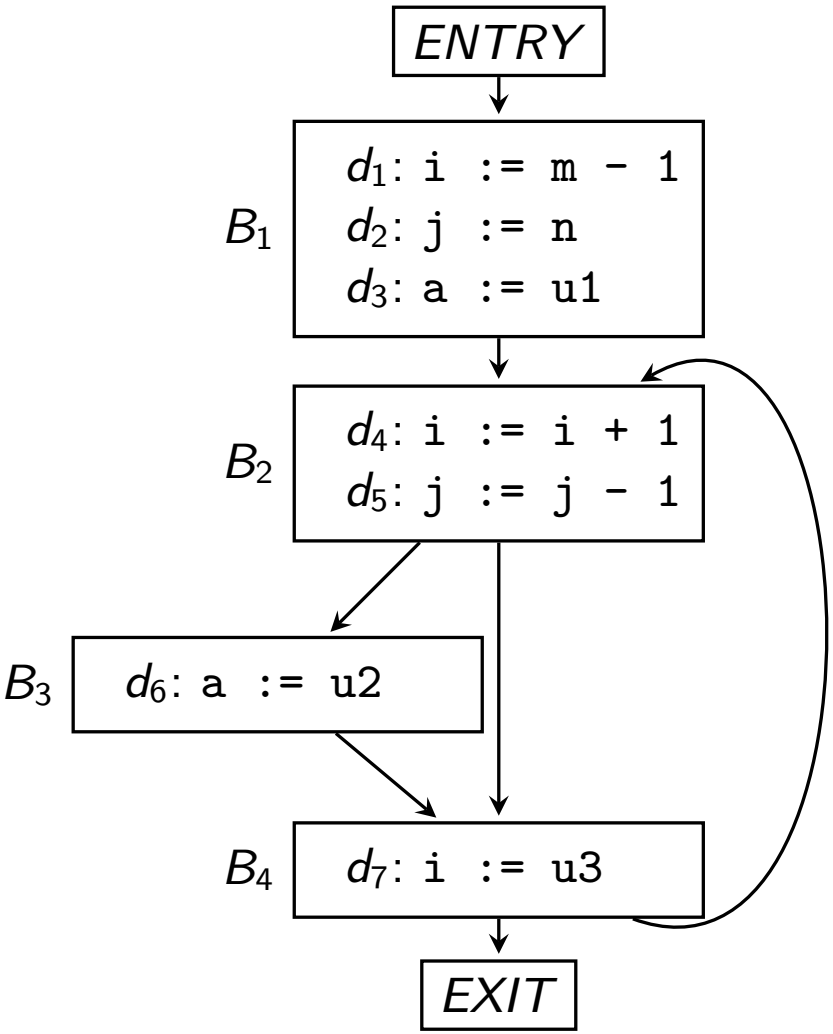


- Todas las de B_1 alcanzan B_2.

UNIVERSIDAD SIMÓN BOLÍVAR

Reaching Definitions

¿Qué queremos determinar?

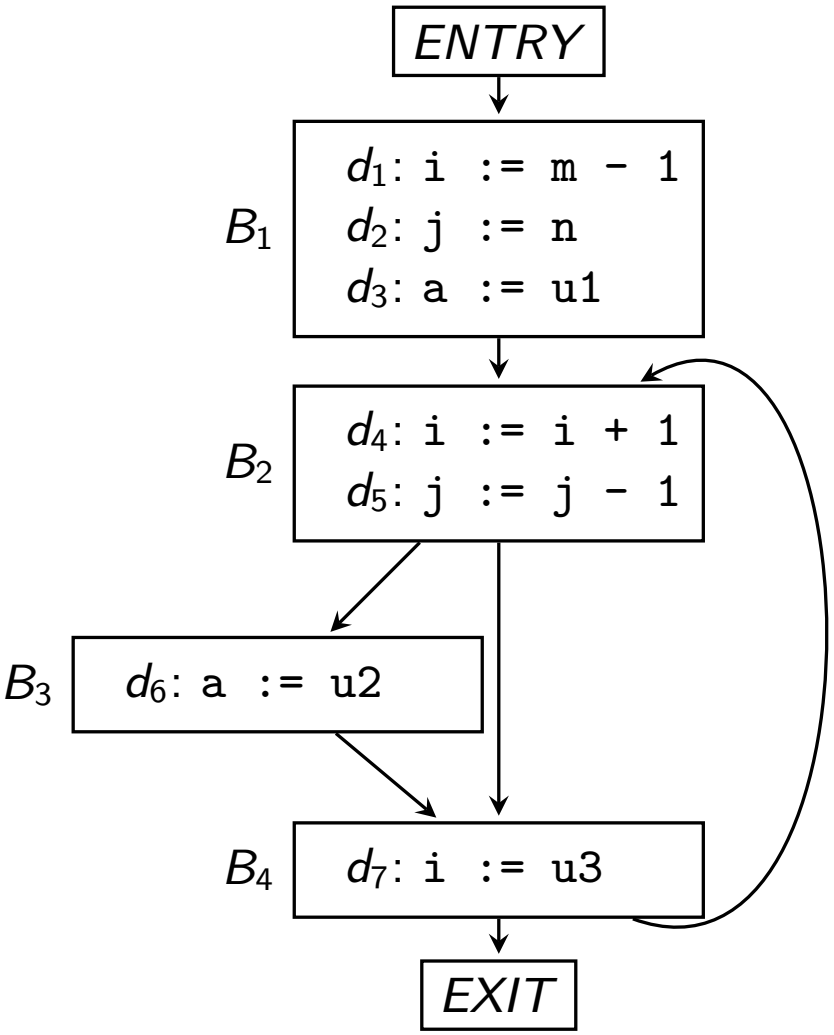


- Todas las de B_1 alcanzan B_2.
- d_5 alcanza el inicio de B_2 – mata a d_2 que no puede alcanzar B_3 ni B_4.

UNIVERSIDAD SIMÓN BOLÍVAR

Reaching Definitions

¿Qué queremos determinar?

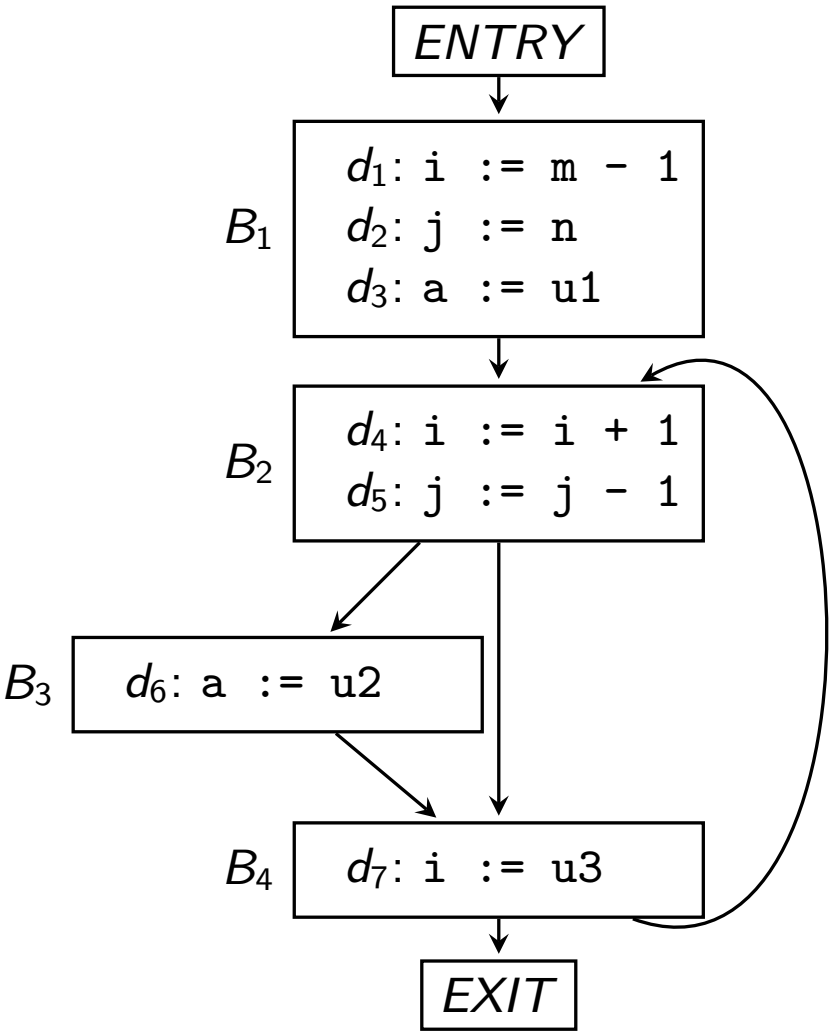


- Todas las de B_1 alcanzan B_2.
- d_5 alcanza el inicio de B_2 – mata a d_2 que no puede alcanzar B_3 ni B_4.
- d_4 no alcanza el inicio de B_2 – es matada por d_7.

UNIVERSIDAD SIMÓN BOLÍVAR

Reaching Definitions

¿Qué queremos determinar?



- Todas las de B_1 alcanzan B_2.
- d_5 alcanza el inicio de B_2 – mata a d_2 que no puede alcanzar B_3 ni B_4.
- d_4 no alcanza el inicio de B_2 – es matada por d_7.
- d_6 alcanza el inicio de B_2.

UNIVERSIDAD SIMÓN BOLÍVAR

Reaching Definitions

¿Cuáles son las restricciones del problema?

$$d\colon \texttt{u := v + w}$$

- Establece a d como definición para `u`.
- Mata cualquier definición previa para `u`.
- El resto de la información se mantiene intacta.
- Establecemos la restricción como Función de Transferencia

$$f_d(x) = gen_d \cup (x - kill_d)$$

donde

- $gen_d = \{d\}$ – definiciones establecidas.
- $kill_d$ – conjunto de definiciones previas para `u`.

UNIVERSIDAD SIMÓN BOLÍVAR

Reaching Definitions

Efecto sobre un Bloque Básico

- Restricción resulta de componer las funciones individuales.
- Para las dos primeras instrucciones tenemos

$$f_1(x) = gen_1 \cup (x - kill_1)$$
$$f_2(x) = gen_2 \cup (x - kill_2)$$

y al componerlas

$$\begin{aligned} f_2(f_1(x)) &= gen_2 \cup (gen_1 \cup (x - kill_1) - kill_2) \\ &= (gen_2 \cup (gen_1 - kill_2)) \cup (x - (kill_1 \cup kill_2)) \end{aligned}$$

UNIVERSIDAD SIMÓN BOLÍVAR

Reaching Definitions

Efecto sobre un Bloque Básico

Extendiendo a todas las instrucciones...

$$f_B(x) = gen_B \cup (x - kill_B)$$

donde

$$kill_B = kill_1 \cup kill_2 \cup \cdots \cup kill_n$$

y

$$\begin{aligned} gen_B = gen_n &\cup (gen_{n-1} - kill_n) \\ &\cup (gen_{n-2} - kill_{n-1} - kill_n) \\ &\cdots \\ &\cup (gen_2 - kill_3 - kill_4 - \cdots - kill_n) \\ &\cup (gen_1 - kill_2 - kill_3 - \cdots - kill_n) \end{aligned}$$

UNIVERSIDAD SIMÓN BOLÍVAR

Reaching Definitions

Bloque Básico es una "gran instrucción"

- gen_B – conjunto de definiciones establecidas por el Bloque Básico efectivas inmediatamente después (*downwards exposed*).
- $kill_B$ – unión de todas las definiciones matadas por alguna instrucción dentro del Bloque Básico.
- Una definición podría aparecer en ambas – la presencia en gen_B tiene precedencia.

UNIVERSIDAD SIMÓN BOLÍVAR

Reaching Definitions

Efectos sobre el Flujo de Control

- Una definición está vigente en un punto del programa cuando existe un camino entre la definición y el punto, i.e.

$$OUT[P] \subseteq IN[B], \text{ si } P \text{ precede a } B$$

- Como una definición no puede alcanzar un punto a menos que haya un camino, es seguro suponer

$$IN[B] = \bigcup_{P \text{ es predecesor de } B} OUT[P]$$

UNIVERSIDAD SIMÓN BOLÍVAR

Reaching Definitions

Las ecuaciones del problema

- Nuestros grafos de flujo siempre tienen *ENTRY* y *EXIT*
  - $f_{ENTRY}(x) = \emptyset$
  - $OUT[ENTRY] = \emptyset$
- Hemos de resolver el sistema de ecuaciones

$$OUT[ENTRY] = \emptyset$$

$$OUT[B] = gen_B \cup (IN[B] - kill_B)$$

$$IN[B] = \bigcup_{P \text{ es predecesor de } B} OUT[P]$$

$$\forall B \neq ENTRY$$

Reaching Definitions

Algoritmo iterativo de punto fijo

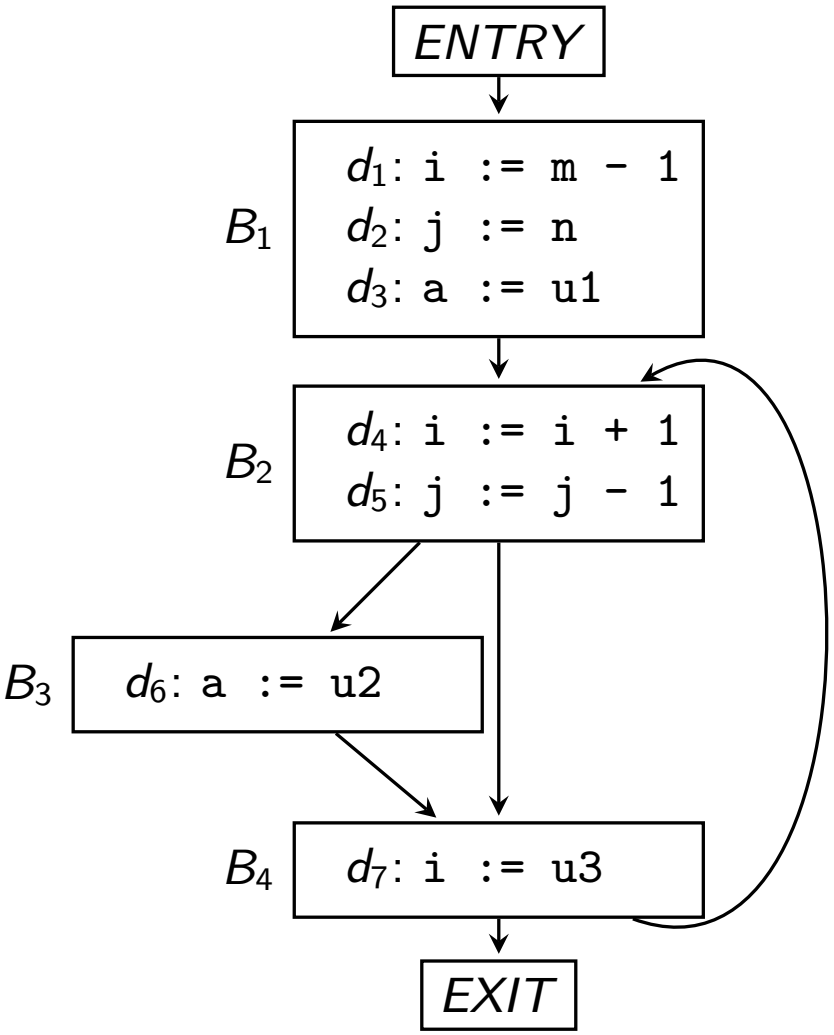
$OUT[ENTRY] \leftarrow \emptyset$
for all $B \neq ENTRY$ **do**
  $OUT[B] \leftarrow \emptyset$
end for
while haya cambios en cualquier OUT **do**
  for all $B \neq ENTRY$ **do**
    $IN[B] \leftarrow \cup_P$ es predecesor de $_B OUT[P]$
    $OUT[B] \leftarrow gen_B \cup (IN[B] - kill_B)$
  end for
end while

- Calcula el *mínimo punto fijo* de las ecuaciones.
- Propaga definiciones tanto como puede antes de ser matadas.

UNIVERSIDAD SIMÓN BOLÍVAR

Reaching Definitions

Inicialización



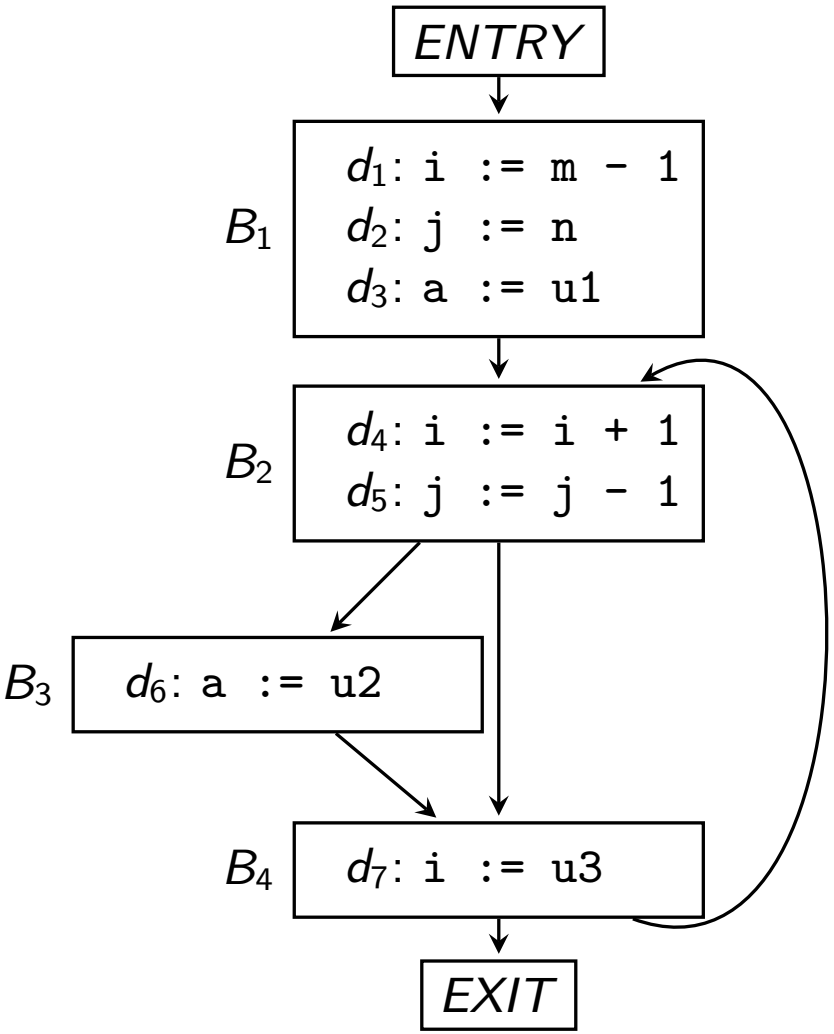
Bloque	*gen*	*kill*
B_1	d_1, d_2, d_3	d_4, d_5, d_6, d_7
B_2	d_4, d_5	d_1, d_2, d_7
B_3	d_6	d_3
B_4	d_7	d_1, d_4

Bloque	*IN*	*OUT*
ENTRY		$\emptyset$
B_1		$\emptyset$
B_2		$\emptyset$
B_3		$\emptyset$
B_4		$\emptyset$
EXIT		$\emptyset$

UNIVERSIDAD SIMÓN BOLÍVAR

Reaching Definitions

Primera iteración



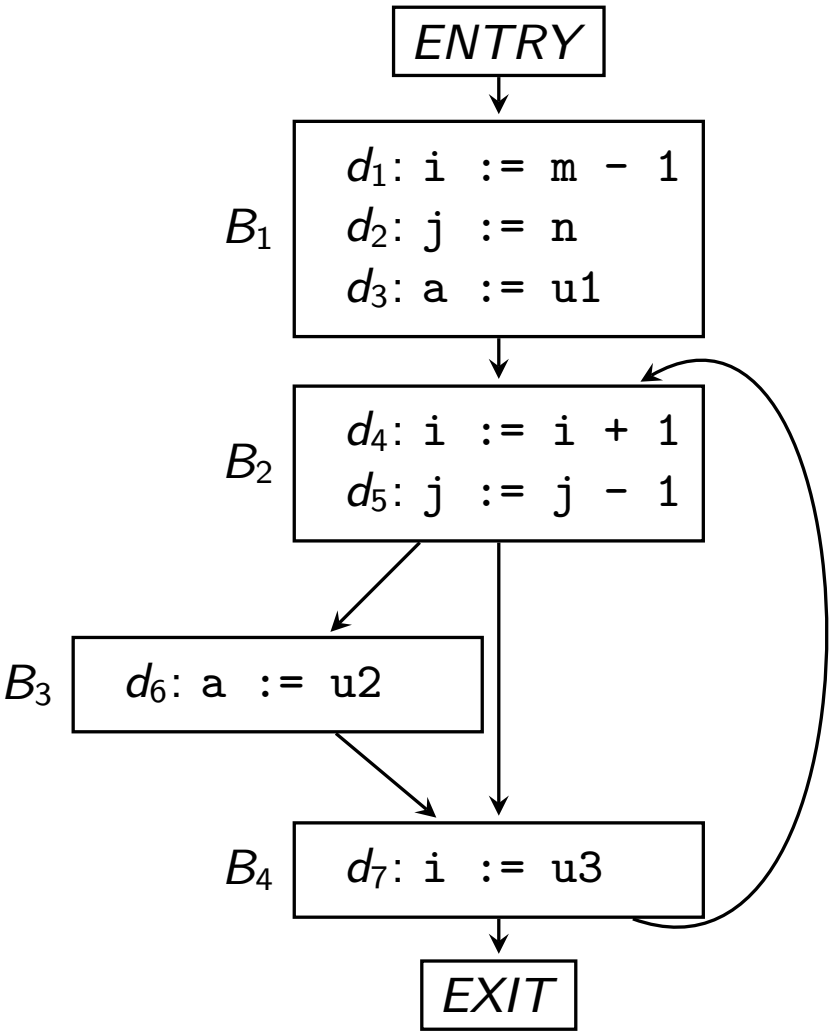
Bloque	*gen*	*kill*
B_1	d_1, d_2, d_3	d_4, d_5, d_6, d_7
B_2	d_4, d_5	d_1, d_2, d_7
B_3	d_6	d_3
B_4	d_7	d_1, d_4

Bloque	*IN*	*OUT*
ENTRY		$\emptyset$
B_1	$\emptyset$	d_1, d_2, d_3
B_2	d_1, d_2, d_3	d_3, d_4, d_5
B_3	d_3, d_4, d_5	d_4, d_5, d_6
B_4	d_3, d_4, d_5, d_6	d_3, d_5, d_6, d_7
EXIT	d_3, d_5, d_6, d_7	d_3, d_5, d_6, d_7

UNIVERSIDAD SIMÓN BOLÍVAR

Reaching Definitions

Segunda iteración



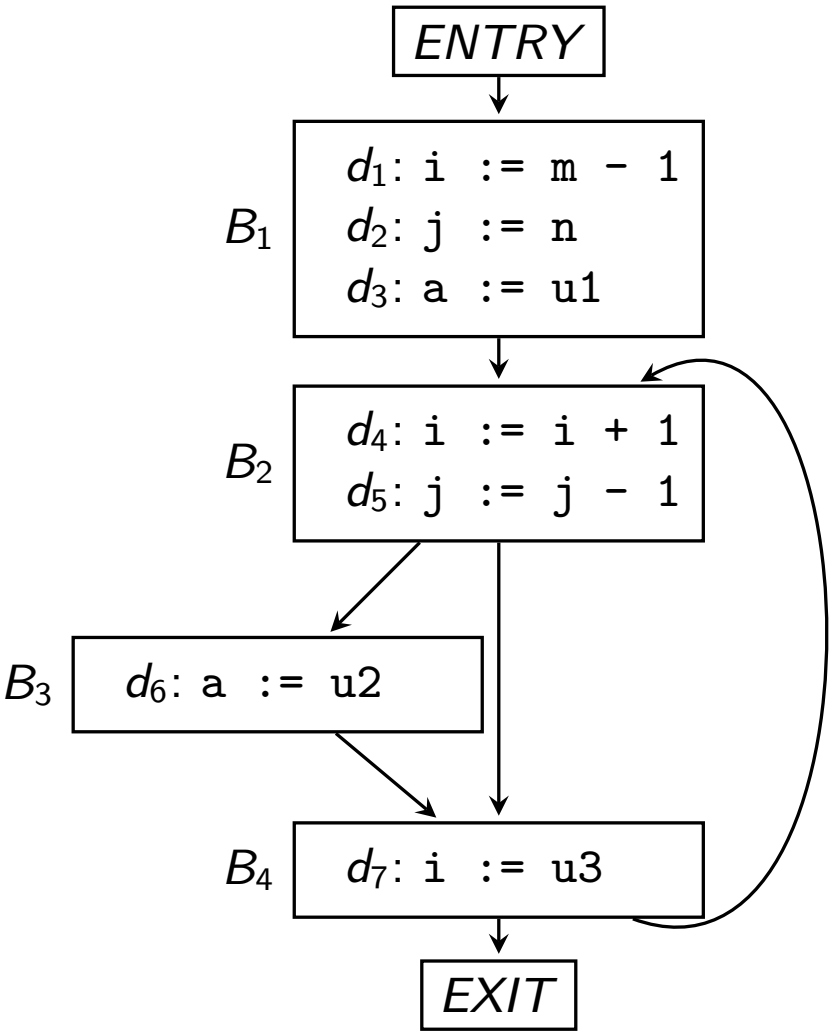
Bloque	*gen*	*kill*
B_1	d_1, d_2, d_3	d_4, d_5, d_6, d_7
B_2	d_4, d_5	d_1, d_2, d_7
B_3	d_6	d_3
B_4	d_7	d_1, d_4

Bloque	*IN*	*OUT*
ENTRY		$\emptyset$
B_1	$\emptyset$	d_1, d_2, d_3
B_2	$d_1, d_2, d_3, d_5, d_6, d_7$	d_3, d_4, d_5, d_6
B_3	d_3, d_4, d_5, d_6	d_4, d_5, d_6
B_4	d_3, d_4, d_5, d_6	d_3, d_5, d_6, d_7
EXIT	d_3, d_5, d_6, d_7	d_3, d_5, d_6, d_7

UNIVERSIDAD SIMÓN BOLÍVAR

Reaching Definitions

Tercera iteración – No hay cambios



Bloque	*gen*	*kill*
B_1	d_1, d_2, d_3	d_4, d_5, d_6, d_7
B_2	d_4, d_5	d_1, d_2, d_7
B_3	d_6	d_3
B_4	d_7	d_1, d_4

Bloque	*IN*	*OUT*
ENTRY		$\emptyset$
B_1	$\emptyset$	d_1, d_2, d_3
B_2	$d_1, d_2, d_3, d_5, d_6, d_7$	d_3, d_4, d_5, d_6
B_3	d_3, d_4, d_5, d_6	d_4, d_5, d_6
B_4	d_3, d_4, d_5, d_6	d_3, d_5, d_6, d_7
EXIT	d_3, d_5, d_6, d_7	d_3, d_5, d_6, d_7

UNIVERSIDAD SIMÓN BOLÍVAR

Funciona, y bien

- Siempre se detiene
  - $OUT[B]$ nunca se reduce – definición agregada nunca se elimina.
  - Número finito de definiciones – eventualmente no habrá agregados.

UNIVERSIDAD SIMÓN BOLÍVAR

Funciona, y bien

- Siempre se detiene
  - $OUT[B]$ nunca se reduce – definición agregada nunca se elimina.
  - Número finito de definiciones – eventualmente no habrá agregados.
- El resultado es el correcto al detenerse
  - Si se detuvo, fue porque no hubo cambios en $OUT[B]$.
  - Si no cambiaron, otra iteración no cambiaría los $IN[B]$.
  - Sin cambios en los $IN[B]$ tampoco cambian los $OUT[B]$.

UNIVERSIDAD SIMÓN BOLÍVAR

Funciona, y bien

- Siempre se detiene
  - $OUT[B]$ nunca se reduce – definición agregada nunca se elimina.
  - Número finito de definiciones – eventualmente no habrá agregados.
- El resultado es el correcto al detenerse
  - Si se detuvo, fue porque no hubo cambios en $OUT[B]$.
  - Si no cambiaron, otra iteración no cambiaría los $IN[B]$.
  - Sin cambios en los $IN[B]$ tampoco cambian los $OUT[B]$.
- El número de bloques es cota superior para el número de iteraciones.
  - Cada iteración hace avanzar definiciones al menos un paso en los caminos, pero usualmente más.
  - Con el orden adecuado para iterar sobre los bloques, bastan **cinco** pasadas en promedio – the law of fives is never wrong.

UNIVERSIDAD SIMÓN BOLÍVAR

Live Variables

Análisis en el otro sentido

- Dados un punto p y una variable x, ¿la variable es usada en algún punto *posterior* a p?
- Necesitamos que $IN[B]$ y $OUT[B]$ representen las variables vivas al entrar y salir de B.
- def_B – variables definidas en B antes de ser usadas en el mismo bloque (se les asignó un valor concreto).
- use_B – variables posiblemente usadas en B antes de ser definidas en el mismo bloque.
- Con esas definiciones
  - Cualquier variable en use_B está viva al entrar a B.
  - Cualquier variable en def_B están muertas al entrar a B.

UNIVERSIDAD SIMÓN BOLÍVAR

Live Variables

Las ecuaciones del problema

- Todas las variables mueren al terminar el programa.
- Una variable está viva en un bloque:
  - Si es usada, ¡duh!
  - Si ya estaba viva y no fue definida en el bloque.
  - Si está viva en alguno de sus sucesores.

$$IN[EXIT] = \emptyset$$

$$IN[B] = use_B \cup (OUT[B] - def_B)$$

$$OUT[B] = \bigcup_{S \text{ es sucesor de } B} IN[S]$$

$$\forall B \neq EXIT$$

UNIVERSIDAD SIMÓN BOLÍVAR

Live Variables

Algoritmo iterativo de punto fijo

$IN[EXIT] \leftarrow \emptyset$
for all $B \neq EXIT$ **do**
  $IN[B] \leftarrow \emptyset$
end for
while haya cambios en cualquier IN **do**
  for all $B \neq EXIT$ **do**
    $OUT[B] \leftarrow \cup_{S \text{ es sucesor de } B} IN[S]$
    $IN[B] \leftarrow use_B \cup (OUT[B] - def_B)$
  end for
end while

Déjà vu!

UNIVERSIDAD SIMÓN BOLÍVAR

¡Se parecen igualito!

- Ambos problemas usan la unión como operador de combinación.
  - Ambos propagan información por los caminos.
  - Sólo importa si *algún* camino cumple la propiedad.

¡Se parecen igualito!

- Ambos problemas usan la unión como operador de combinación.
  - Ambos propagan información por los caminos.
  - Sólo importa si *algún* camino cumple la propiedad.
- La *dirección* particular establece la manipulación
  - Caso base – usa *OUT* hacia adelante; usa *IN* hacia atrás.
  - Caso inductivo – estudia *OUT* hacia adelante; estudia *IN* hacia atrás.
- Los roles de *gen* y *kill* son reemplazados por *use* y *def* .

UNIVERSIDAD SIMÓN BOLÍVAR

¡Se parecen igualito!

- Ambos problemas usan la unión como operador de combinación.
  - Ambos propagan información por los caminos.
  - Sólo importa si *algún* camino cumple la propiedad.
- La *dirección* particular establece la manipulación
  - Caso base – usa *OUT* hacia adelante; usa *IN* hacia atrás.
  - Caso inductivo – estudia *OUT* hacia adelante; estudia *IN* hacia atrás.
- Los roles de *gen* y *kill* son reemplazados por *use* y *def* .

Implantar el algoritmo de manera genérica, aprovechando polimorfismo para "engancharse"

UNIVERSIDAD SIMÓN BOLÍVAR

Bibliografía

- [*Aho*]
  - Sección 9.2
  - Ejercicios 9.2.1 a 9.2.11

UNIVERSIDAD SIMÓN BOLÍVAR