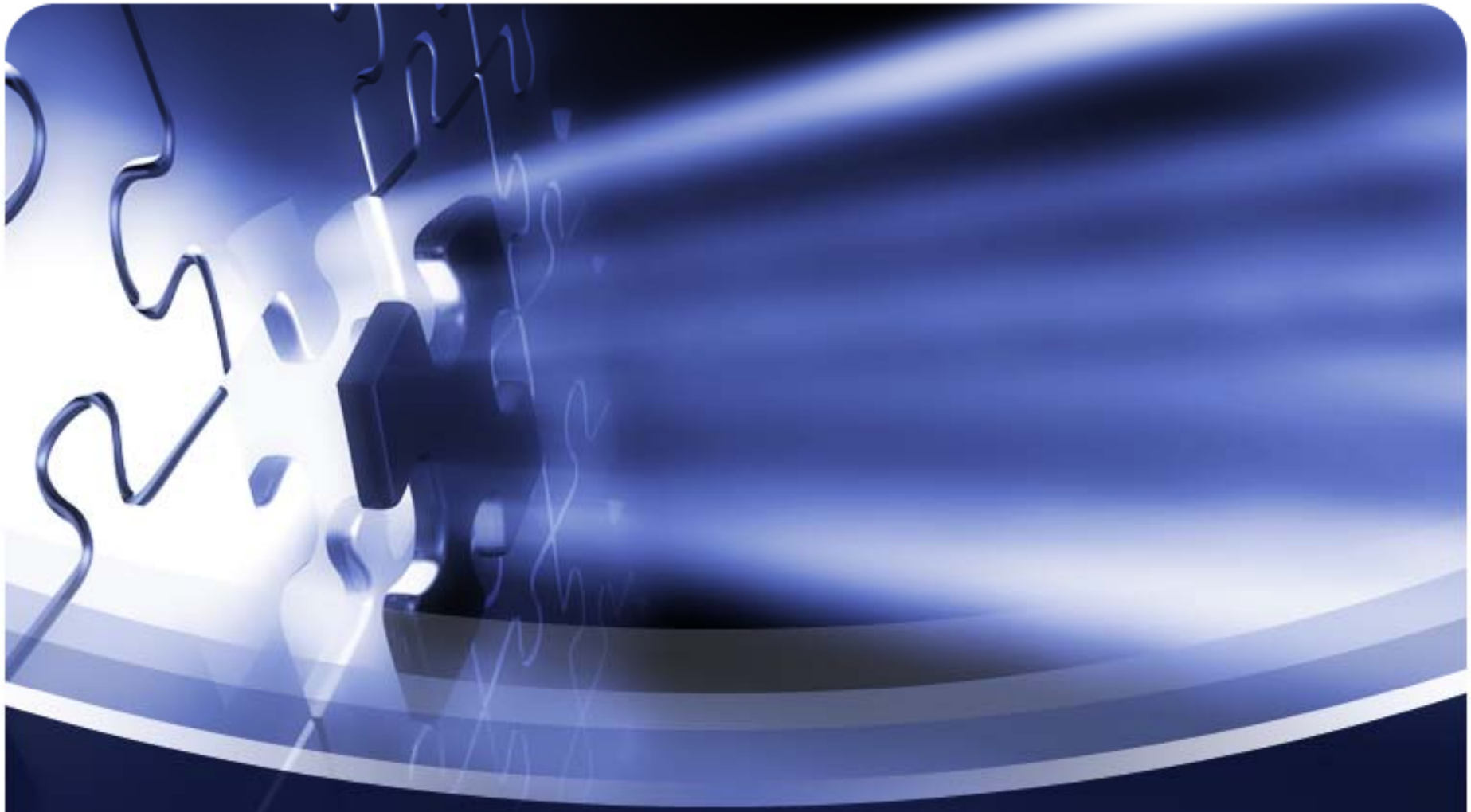


Fuzzy Logic & Computational Geometry

Simon Coupland & Robert John

Fuzzy Logic & Computational Geometry

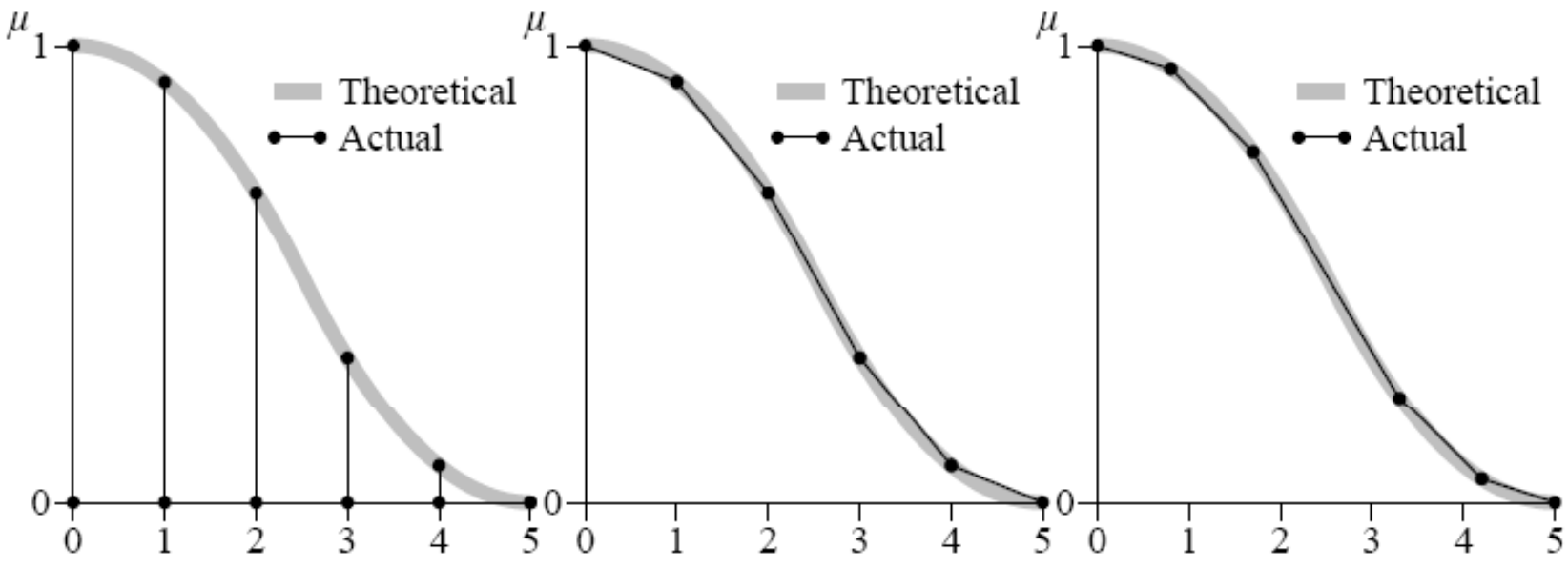
- بررسی فازی بودن از دیدگاه هندسه ی محاسباتی
- معرفی یک نمایان مجموعه ی فازی هندسی
- بحث بر روی دقت بیشتر این نمایان
- معرفی عملگرهای AND و OR بر روی این نمایان
  - با یک دامنه ی پیوسته،
  - و دقتی که فقط توسط نمایش اعداد حقیقی در رایانه زیر سؤال می رود.

Fuzzy Logic & Computational Geometry

- انواع نمایش مجموعه های فازی در رایانه:
  - Singleton: نقاطی از فضای گسسته و مقادیرشان
  - NRC: مجموعه ای از نقاط پیوسته توسط خطوط مقادیرشان در بازه های تعریف شده ی طبیعی (Polyline)
  - Geometric: مجموعه ای از نقاط پیوسته توسطِ خطوط مقادیرشان در بازه های تعریف شده ی نه لزوما طبیعی و نه لزوما هم فاصله (Polyline)

Fuzzy Logic & Computational Geometry

- انواع نمایش مجموعه های فازی در رایانه:
  - Singleton –
  - NRC –
  - Geometric –



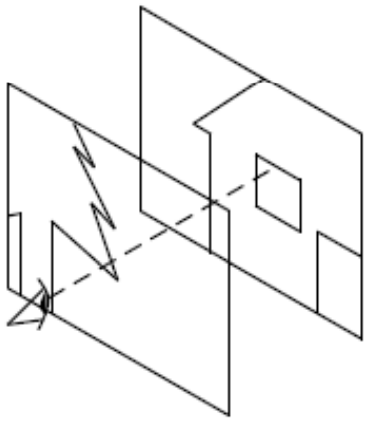
Fuzzy Logic & Computational Geometry

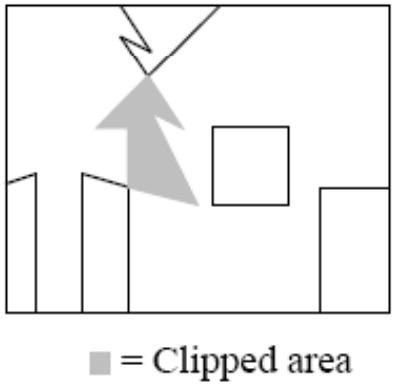
- نمایش مجموعه های فازی در حوزه ی هندسه ی محاسباتی
  - مقادیر وابستگی خطوط بین ۰ و ۱
- مقایسه ی سه نوع نمایش

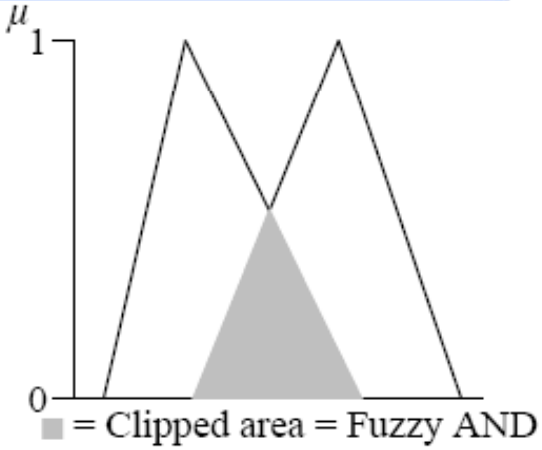
Representation	Centroid (3 s.f.)	Actual Centroid (3 s.f.)
Singleton	1.70	1.73
NRC	1.70	1.73
Geometric	1.72	1.73

Fuzzy Logic & Computational Geometry

- یک مسأله ی مهم در گرافیک کامپیوتری، برش زدن (Clip) یک کثیرالاضلاع در برابر دیگری است:
  - Hidden Surface removal
  - برای مثال: بریدن کثیرالاضلاع خانه در برابر درخت
  - به زبان فازی، فضای بریده شده اشتراک دو مجموعه ی فازی درخت و خانه است.







Fuzzy Logic & Computational Geometry

- الگوریتم های مختلفی برای Clipping وجود دارد:
  - انتخاب ما الگوریتم Weiler - Atherton
  - نیازمند یک الگوریتم دیگر برای جاروب سطوح (Plane Sweep)
  - در راستای یافتن نقاط تقاطع سطوح
    - در اینجا همان Polyline های ما!
  - بررسی ما تنها برای دو Polyline
    - نیازمند تغییر الگوریتم برای ایجاد تناسب لازم با مسأله ی ما
  - برای ایجاد امکان پیمایش Polylineها:
    - ذخیره ی رئوس هر Polyline به صورت ساعتگرد از یک نقطه ی شروع در یک لیست

Fuzzy Logic & Computational Geometry

- الگوریتم Weiler – Atherton برای محاسبه ی عملگر AND فازی برای دو Polyline نظیر A و B:

I. تمام نقاط تقاطع دو Polyline را پیدا کن.

II. لیست رئوس A و B را با در نظر گرفتن نقاط تقاطع ایجاد کن.

III. نقطه ی شروع را رأس سر لیست با مختصات X بیشتر قرار بده؛ این نقطه را در خروجی قرار بده.

IV. پیمایش لیست را آغاز کن؛ هر جا به نقطه ی تقاطع رسیدی، لیست ها را عوض کن.

V. مرحله ی IV را تا جایی ادامه بده که یکی از دو لیست به اتمام برسند.

Fuzzy Logic & Computational Geometry

- الگوریتم Weiler – Atherton برای محاسبه ی عملگر OR فازی برای دو Polyline نظیر A و B:

I. تمام نقاط تقاطع دو Polyline را پیدا کن.

II. لیست رئوس A و B را با در نظر گرفتن نقاط تقاطع ایجاد کن.

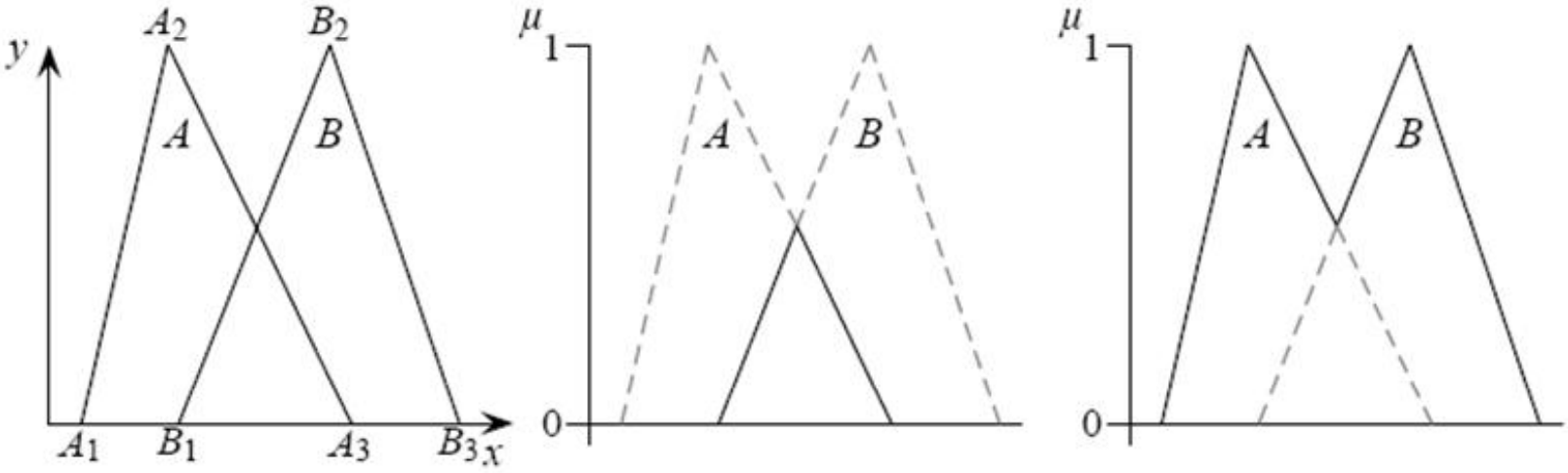
III. نقطه ی شروع را رأس سر لیست با مختصات X کمتر قرار بده؛ این نقطه را در خروجی قرار بده.

IV. پیمایش لیست را آغاز کن؛ هر جا به نقطه ی تقاطع رسیدی، لیست ها را عوض کن.

V. مرحله ی IV را تا جایی ادامه بده که یکی از دو لیست به اتمام برسند.

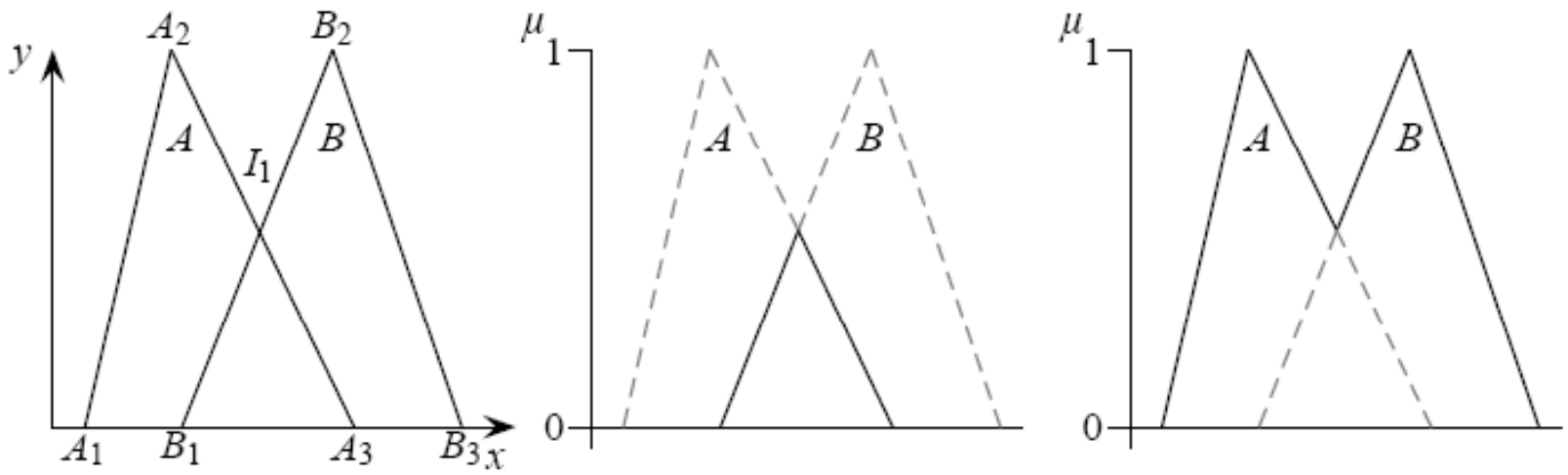
Fuzzy Logic & Computational Geometry

- مثالی برای محاسبه ی عملگرهای AND و OR فازی برای دو Polyline نظیر A و B:



Fuzzy Logic & Computational Geometry

- محاسبه ی عملگر AND فازی برای دو Polyline نظیر A و B:
  I. تمام نقاط تقاطع دو Polyline را پیدا کن.



Fuzzy Logic & Computational Geometry

- محاسبه ی عملگر AND فازی برای دو Polyline نظیر A و B:

II. لیست رئوس A و B را با در نظر گرفتن نقاط تقاطع ایجاد کن.

Vertex List	Value				
A	A_1	A_2	I_1	A_3	
B		B_1	I_1	B_2	B_3

Fuzzy Logic & Computational Geometry

- محاسبه ی عملگر AND فازی برای دو Polyline نظیر A و B:

III. نقطه ی شروع را رأس سر لیست با مختصات X بیشتر قرار بده؛ این نقطه را در خروجی قرار بده.

OUTPUT := <B_1>

Fuzzy Logic & Computational Geometry

- محاسبه ی عملگر AND فازی برای دو Polyline نظیر A و B:

IV. پیمایش لیست را آغاز کن؛ هر جا به نقطه ی تقاطع رسیدی، لیست ها را عوض کن. (اجرای اول)

OUTPUT := $<B_1, I_1>$

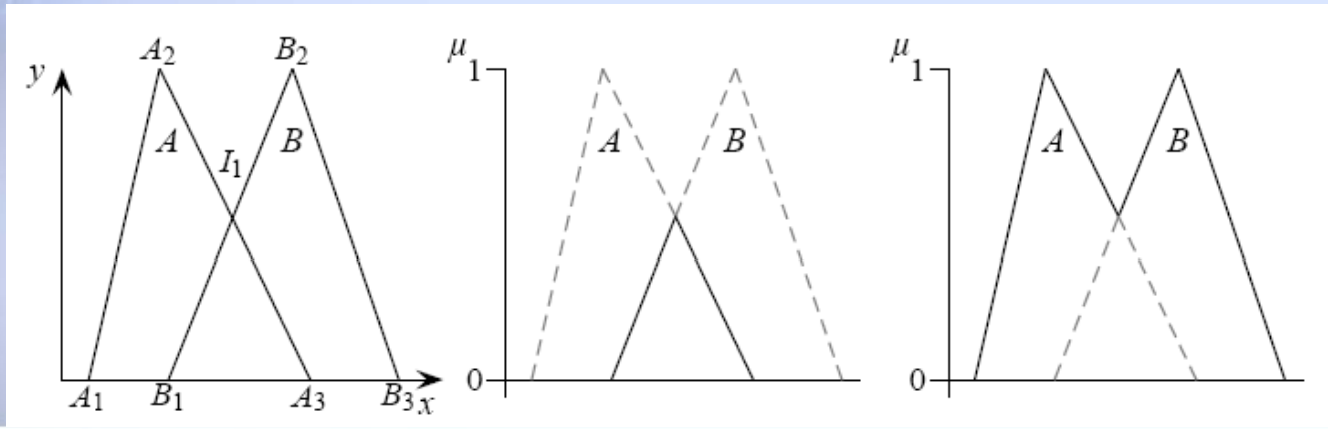
Fuzzy Logic & Computational Geometry

- محاسبه ی عملگر AND فازی برای دو Polyline نظیر A و B:

IV. پیمایش لیست را آغاز کن؛ هر جا به نقطه ی تقاطع رسیدی، لیست ها را عوض کن. (اجرای دوم)

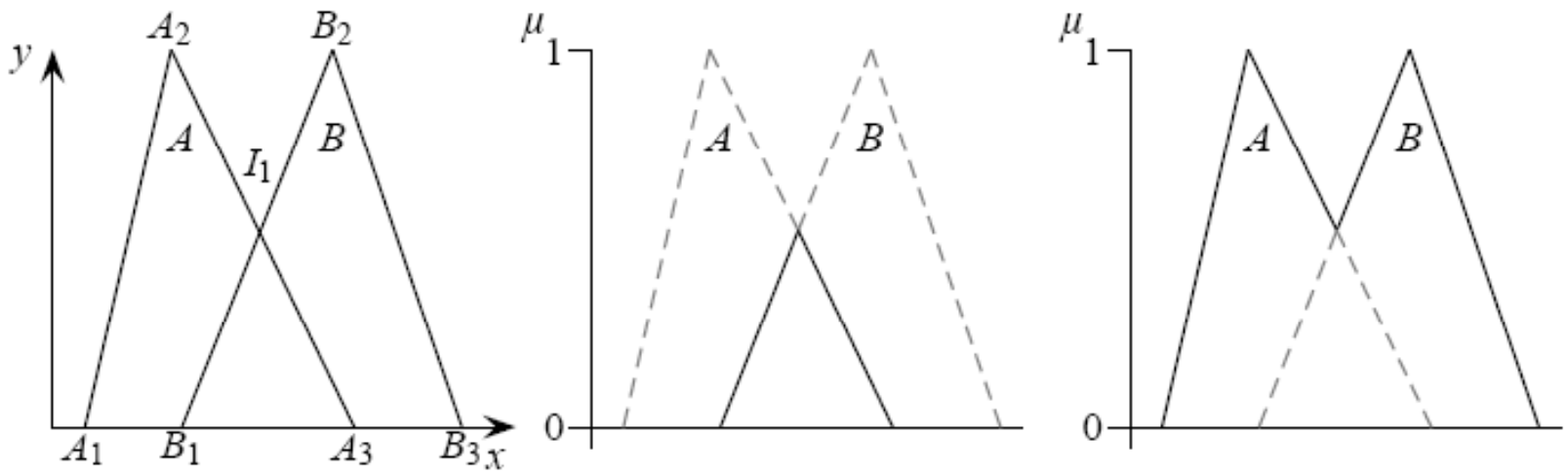
OUTPUT := <B_1, I_1, A_3>

انتهای لیست مربوط به A



Fuzzy Logic & Computational Geometry

- محاسبه ی عملگر OR فازی برای دو Polyline نظیر A و B:
  I. تمام نقاط تقاطع دو Polyline را پیدا کن.



Fuzzy Logic & Computational Geometry

- محاسبه ی عملگر OR فازی برای دو Polyline نظیر A و B:

II. لیست رئوس A و B را با در نظر گرفتن نقاط تقاطع ایجاد کن.

Vertex List	Value				
A	A_1	A_2	I_1	A_3	
B		B_1	I_1	B_2	B_3

Fuzzy Logic & Computational Geometry

- محاسبه ی عملگر OR فازی برای دو Polyline نظیر A و B:

III. نقطه ی شروع را رأس سر لیست با مختصات X کمتر قرار بده؛ این نقطه را در خروجی قرار بده.

OUTPUT := $<A_1>$

Fuzzy Logic & Computational Geometry

- محاسبه ی عملگر OR فازی برای دو Polyline نظیر A و B:

IV. پیمایش لیست را آغاز کن؛ هر جا به نقطه ی تقاطع رسیدی، لیست ها را عوض کن. (اجرای اول)

OUTPUT := $<A_1, A_2>$

Fuzzy Logic & Computational Geometry

- محاسبه ی عملگر OR فازی برای دو Polyline نظیر A و B:

IV. پیمایش لیست را آغاز کن؛ هر جا به نقطه ی تقاطع رسیدی، لیست ها را عوض کن. (اجرای دوم)

OUTPUT := $<A_1, A_2, I_1>$

Fuzzy Logic & Computational Geometry

- محاسبه ی عملگر OR فازی برای دو Polyline نظیر A و B:

IV. پیمایش لیست را آغاز کن؛ هر جا به نقطه ی تقاطع رسیدی، لیست ها را عوض کن. (اجرای سوم)

OUTPUT := $<A_1, A_2, I_1, B_2>$

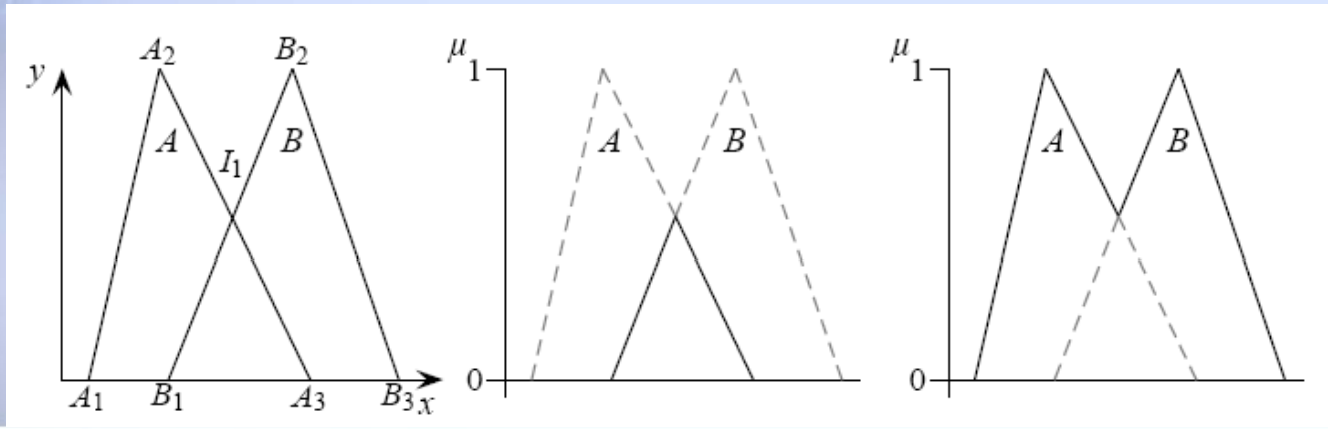
Fuzzy Logic & Computational Geometry

- محاسبه ی عملگر OR فازی برای دو Polyline نظیر A و B:

IV. پیمایش لیست را آغاز کن؛ هر جا به نقطه ی تقاطع رسیدی، لیست ها را عوض کن. (اجرای چهارم)

OUTPUT := <A_1, A_2, I_1, B_2, B_3>

انتهای لیست مربوط به B



Fuzzy Logic & Computational Geometry

- الگوریتمی برای جاروب سطوح (Plane Sweep): برای مرحله ی اول از دو الگوریتم مورد بحث لازم است.
  - راهکار اولیه یک الگوریتم Brute-Force است که امکان تقاطع هر پاره خط متعلق به یک کثیرالاضلاع را با تمام پاره خط های کثیرالاضلاع دیگر بررسی می کند.
    - O(2(M+N))
  - الگوریتم Bentley – Ottmann که یک خط فرضی عمودی را در راستای محور X حرکت می دهد و تنها برای پاره خطوطی بررسی را انجام می دهد که نقطه ی شروع شان را پیموده ولی نقطه ی پایان شان را نه. تمام نقاط سمت چپ خط مرتفع شده اند.
    - O((M+N)log(M+N))
    - ساختمان داده ی مورد نیاز: یک صف رویداد ها، دو رجیستر، یک لیست برای پاسخ

Fuzzy Logic & Computational Geometry

- الگوریتم Bentley – Ottmann:

I. تمام نقاط تمام پاره خط ها را در صف رویدادها قرار بده و آن را مرتب کن.

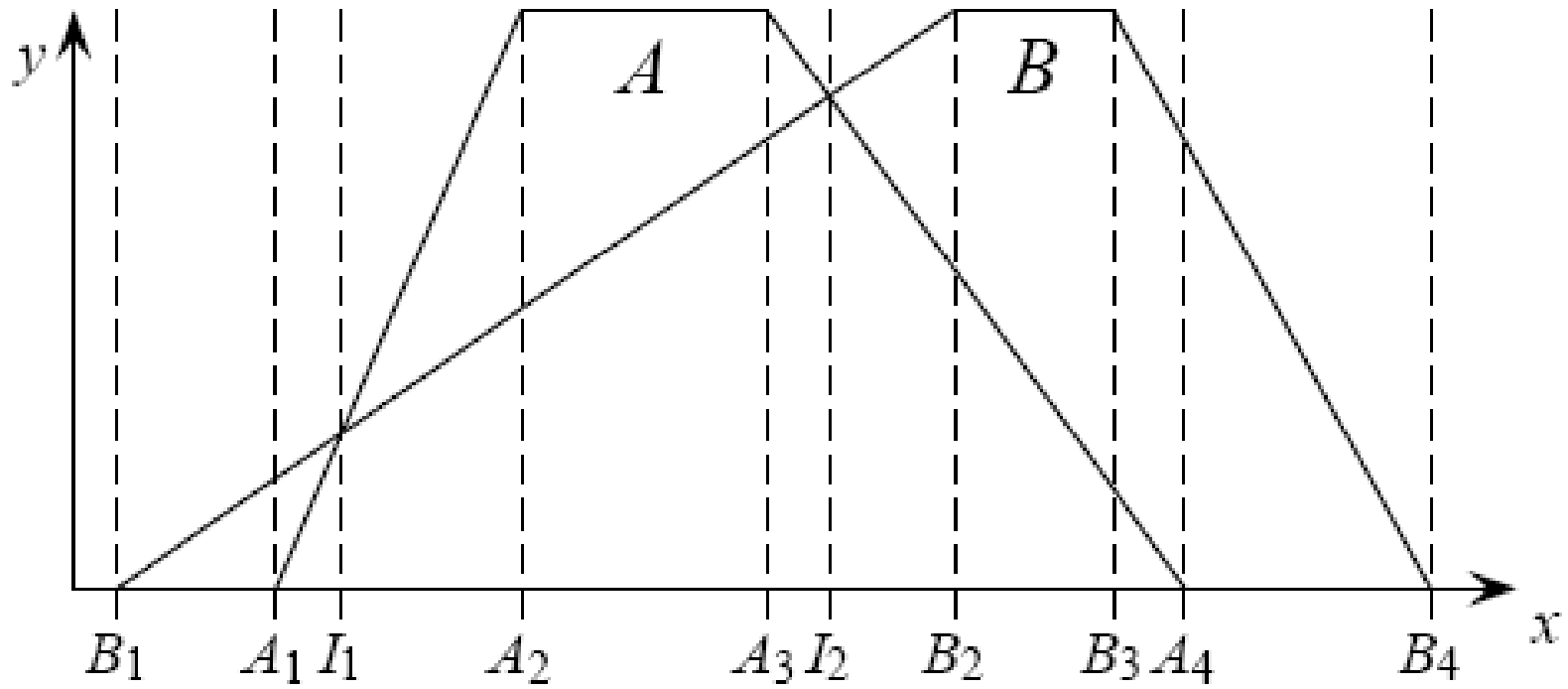
II. اولین عنصر صف را فراخوانی کن. رجیستر مرتبط را با پاره ط مورد نظر مقداردهی کن.

III. اگر پاره خط مورد نظر با پاره خط رجیستر دیگر تقاطع داشت، آن را در لیست خروجی قرار بده. حال صف رویدادها را دوباره مرتب کن.

IV. مراحل ۲ و ۳ را تا جایی که صف رویدادها خالی شود تکرار کن.

Fuzzy Logic & Computational Geometry

- مثالی برای محاسبه ی نقاط تقاطع با کمک الگوریتم Bentley – Ottmann:



Fuzzy Logic & Computational Geometry

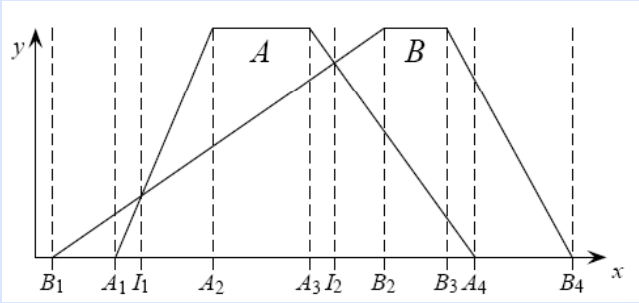
- مثالی برای محاسبه ی نقاط تقاطع با کمک الگوریتم – Bentley Ottmann:

Step 1:

Variable	Value
Event queue	$B_1, A_1, A_2, A_3, B_2, B_3, A_4, B_4$
Line A register	NULL
Line B register	NULL
Intersection Points	

Steps 2 and 3 (first iteration):

Variable	Value
Event queue	$A_1, A_2, A_3, B_2, B_3, A_4, B_4$
Line A register	NULL
Line B register	$\lvert B_1, B_2 \rvert$
Intersection Points	

Steps 2 and 3 (second iteration):

Variable	Value
Event queue	$I_1, A_2, A_3, B_2, B_3, A_4, B_4$
Line A register	$\lvert A_1, A_2 \rvert$
Line B register	$\lvert B_1, B_2 \rvert$
Intersection Points	I_1