Operace s uzavřenými oblastmi v GIS

Booleovské operace s uzavřenými oblastmi. Offset polygonu. Minkowského suma.

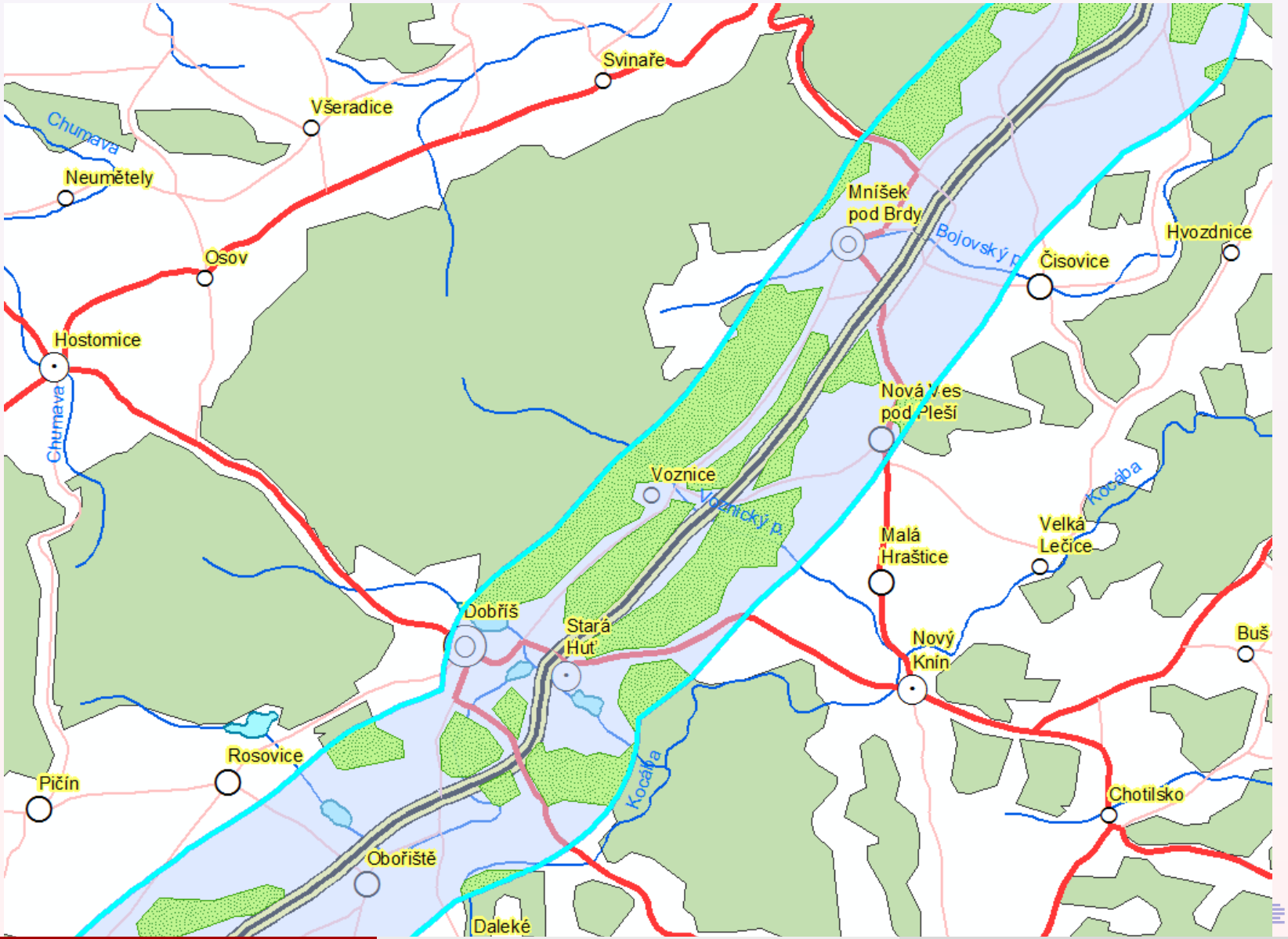
Tomáš Bayer | bayertom@natur.cuni.cz

Katedra aplikované geoinformatiky a kartografie. Přírodovědecká fakulta UK.

Obsah přednášky

1. Ukázka použití offsetu polygonu a množinových operací
2. Množinové operace, formulace problému
3. Množinové operace s nekonvexními regiony
   - Množinové operace s využitím ohodnocení hran
   - Algoritmus konstrukce
4. Offset oblasti C
   - Exaktní konstrukce offsetu
   - Minkowského suma
   - Ukázky Minkowského sumy různých množin
   - Konstrukce offsetu s využitím Minkowského sumy
   - Algoritmus pro konstrukci Minkowského sumy

1. Nalezení lesů ležících do 2 km od dálnice



2. Formulace problému

Dáno:	Dvojice uzavřených, ohraničených regionů A, B v $\mathbb{R}^2$.
Hledáme:	$C = A \cap B$, $C = A \cup B$, $C = A \cap \overline{B}$, $C = B \cap \overline{A}$,

Základní analytický nástroj GIS.

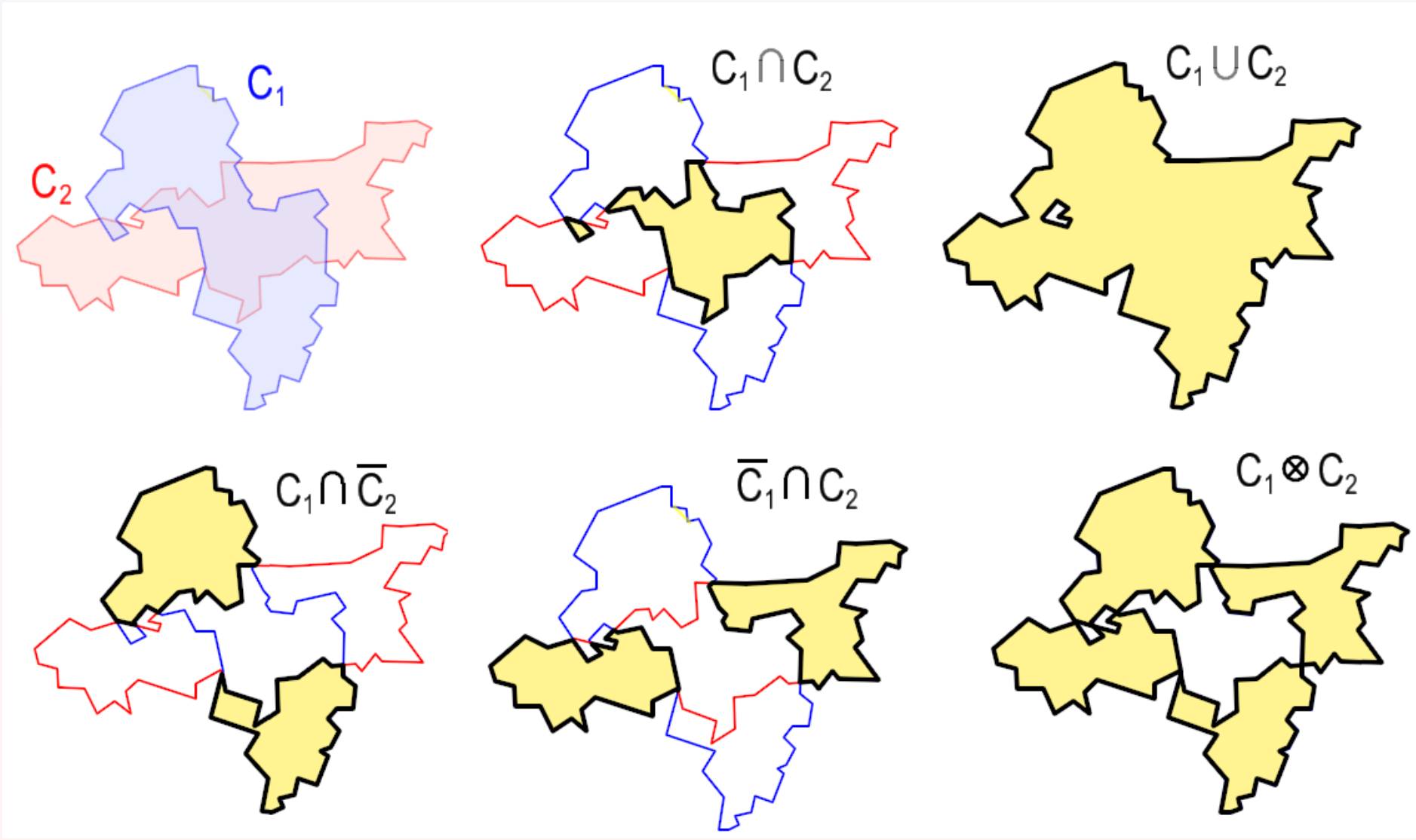
Syntéza a integrace dat z různých zdrojů.

Přehled množinových operací:

- Sjednocení (Union): $C = A \cup B$.
- Průnik (Intersection): $C = A \cap B$.
- Rozdíl (Difference): $C = A \cap \overline{B}$ resp. $C = B \cap \overline{A}$.
- Symmetric difference (XOR): $C = A \triangle B = (A \cup B) \cap \left(\overline{A \cap B}\right)$.

V geoinformatické praxi implementovány pouze operace 1-3.

3. Ilustrace množinových operací



4. Vlastnosti množinových operací

Komutativita:

$$A \cup B = B \cup A,$$
$$A \cap B = B \cap A,$$
$$A \cap \overline{B} \neq B \cap \overline{A},$$
$$(A \cup B) \cap \left(\overline{A \cap B}\right) = (B \cup A) \cap \left(\overline{B \cap A}\right)$$

Asociativita

$$A \cup (B \cup C) = (A \cup B) \cup C,$$
$$A \cap (B \cap C) = (A \cap B) \cap C,$$
$$A \cap (\overline{B \cap \overline{C}}) \neq (A \cap \overline{B}) \cap \overline{C},$$
$$A \triangle (B \triangle C) = (A \triangle B) \triangle C,$$

kde

$$A \triangle (B \triangle C) = (A \cup [(B \cup C) \cap \overline{(B \cap C)}]) \cap \overline{(A \cap [(B \cup C) \cap \overline{(B \cap C)}]},$$
$$(A \triangle B) \triangle C = ([(A \cup B) \cap (\overline{A \cap B})] \cup C) \cap \overline{([(A \cup B) \cap (\overline{A \cap B})] \cap C)}.$$

5. Množinové operace s nekonvexními polygony

Úloha běžně řešená v SW GIS.

Vstupem konvexní či nekonvexní polygony.

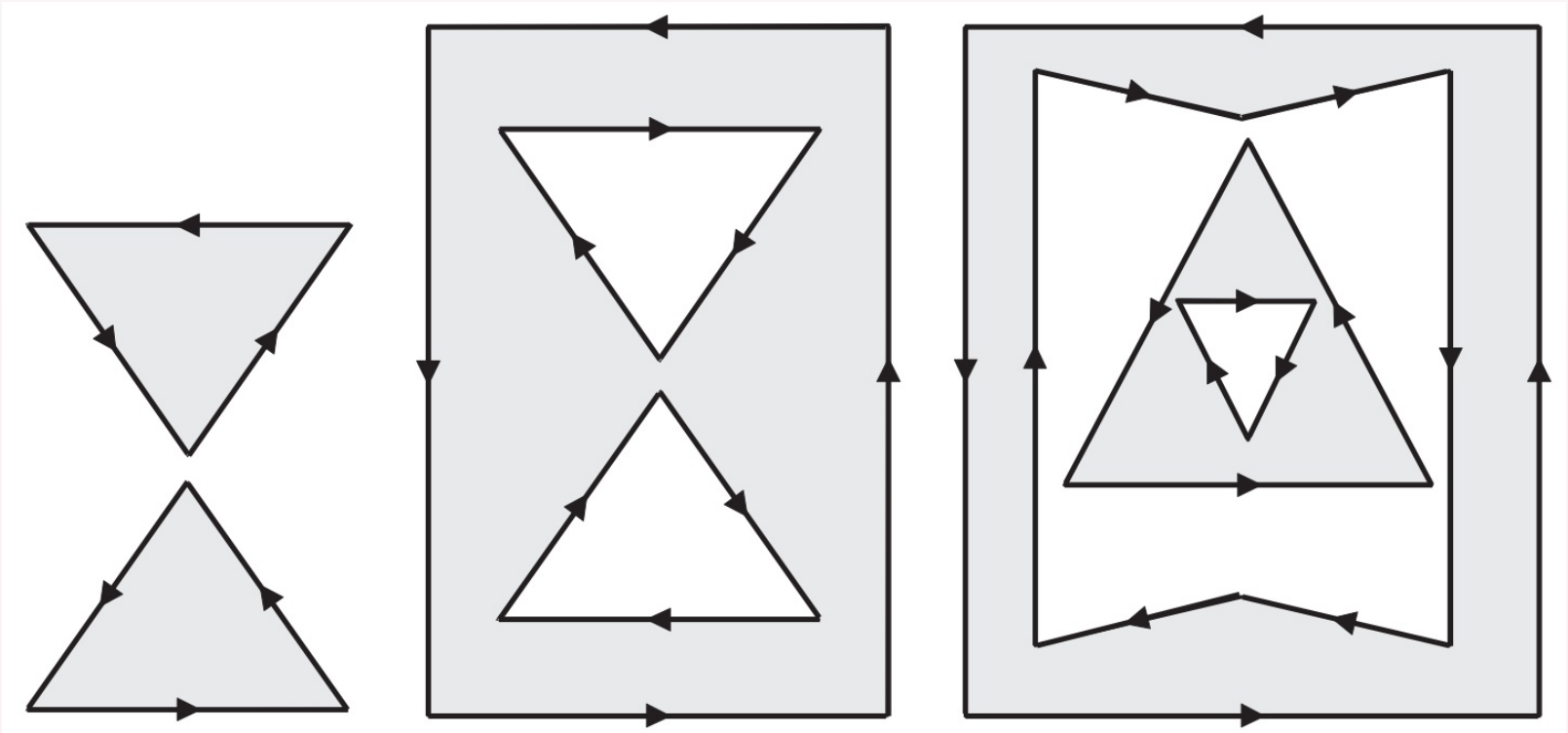
Požadavek na stabilitu a výpočetní efektivitu.

Různé strategie a přístupy řešení:

- *Dekompozice na elementární geometrické operace* (Smith and Dogson, 2006)
  Hierarchická dekompozice (6 úrovní), celkem 16 operací.
  Výsledek topologicky korektní.
- *Simplicial chains* (Rivero, 2000, Peng et al, 2005, ...)
  Rozklad oblastí na řetězce, nad nimi implementovány množinové operace.
  Složitější implementace.
- *Rekonstrukce s využitím vzájemné polohy hran*
  Často používaná metoda (Greiner, 1998, Martinez et al, 2009).
  Hrany děleny na vnitřní/vnější/boundary.
  Nevýhoda: robustní metoda řešení PLP.

6. Datový model

Uzavřené oblasti mohou obsahovat otvory (opačná orientace).



7. Množinové operace, vzájemná poloha hran

Použití pro nekonvexní oblasti včetně otvorů.

Uzavřené oblasti $A = \{p_i\}_{i=1}^{n}$, $B = \{q_j\}_{j=1}^{m}$.
Složitost $O(m * n)$.

Výsledkem operace

$$C = A \otimes B,$$

prázdná množina, 0D entita, 1D entita, 2D entita nebo jejich kombinace.

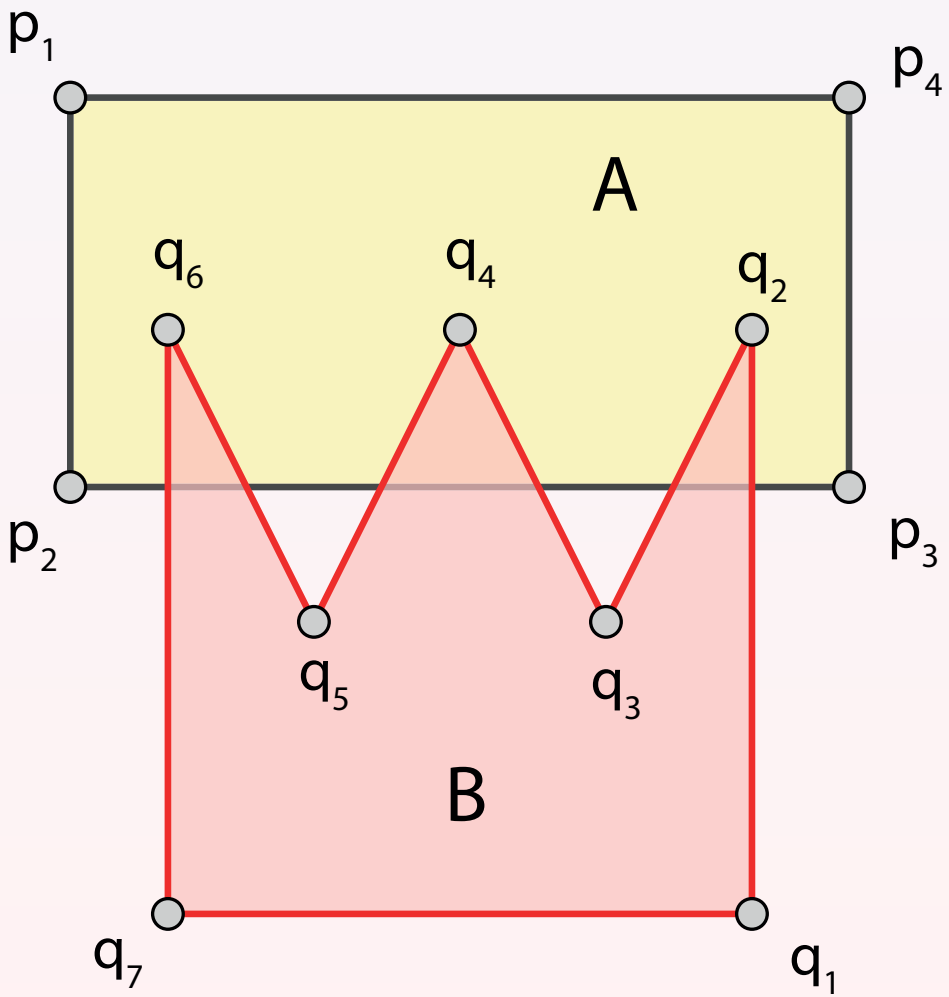
A, B reprezentovány kruhovými seznamy.
CCW orientace.

Fáze algoritmu:

A) Výpočet průsečíků A, B + setřídění
B) Update A, B.
C) Ohodnocení vrcholů A, B dle pozice vůči B, A.
D) Výběr vrcholů dle ohodnocení.
E) Vytvoření hran.
F) Spojení hran do oblastí.

8. Ukázka, oblasti A, B

A	p_1	p_2	p_3	p_4			
B	q_1	q_2	q_3	q_4	q_5	q_6	q_7



9. Výpočet průsečíků A, B

Oblasti A, B mají k průsečíků b_{ij},

$$b_{ij} = e_i \cap e'_j, \qquad e_i = (p_i, p_{i+1}), e'_j = (j, q_{j+1}).$$

Metoda výpočtu průsečíků:

1) Naivní algoritmus

Složitostí $O(m \cdot n)$, test všech kombinací $e_i \cap e'_j$.

2) Line Sweep (Bentley-Ottman).

Složitostí $\theta(n \cdot \log(n))$, obtížnější implementace.

Výpočet průsečíku b_{ij} segmentů e_i, e'_j:

$$b_{ij} = p_i + \alpha \overrightarrow{u} = p_j + \beta \overrightarrow{v},$$

směrové vektory $\overrightarrow{u}, \overrightarrow{v}, \overrightarrow{w}$

$$\overrightarrow{u} = (x_{i+1} - x_i, y_{i+1} - y_i), \qquad \overrightarrow{v} = (x_{j+1} - x_j, y_{j+1} - y_j), \qquad \overrightarrow{w} = (x_i - x_j, y_i - y_j).$$

Parametry α, β

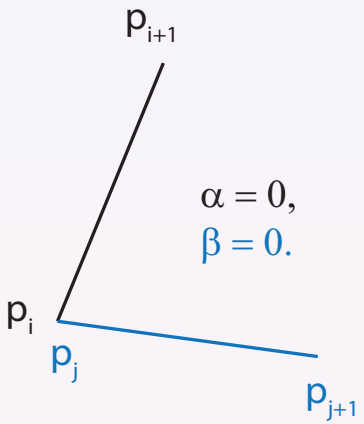
$$\alpha = \frac{k_1}{k_3}, \qquad \beta = \frac{k_2}{k_3}, \qquad k_1 = v_x w_y - v_y w_x, \qquad k_2 = u_x w_y - u_y w_x, \qquad k_3 = v_y u_x - v_x u_y.$$

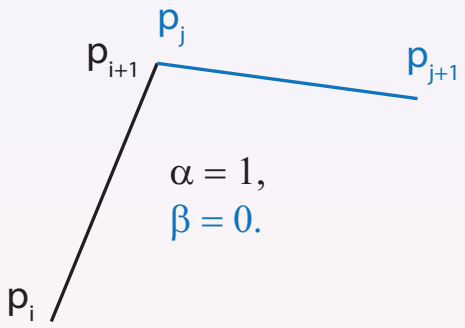
$k_1 = 0$, $k_2 = 0$, $k_3 = 0$: e_i, e'_j kolineární.

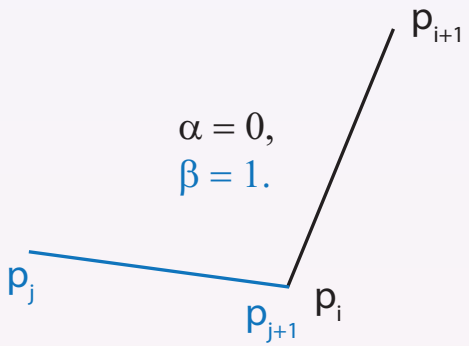
$k_1 = 0$ a $k_2 = 0$: e_i, e'_j rovnoběžné.

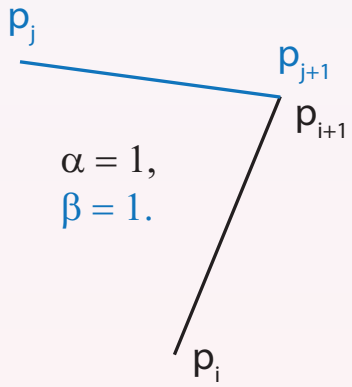
$\alpha \in \langle 0, 1 \rangle$ a $\beta \in \langle 0, 1 \rangle$: $\exists b_{ij}$, jinak mimoběžné.

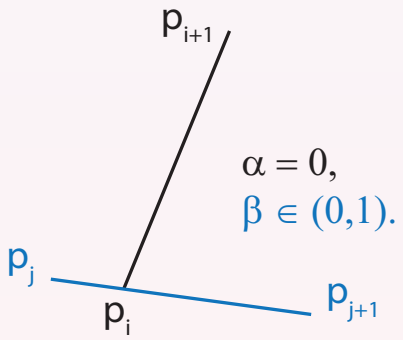
10. Ukázky průsečíků segmentů

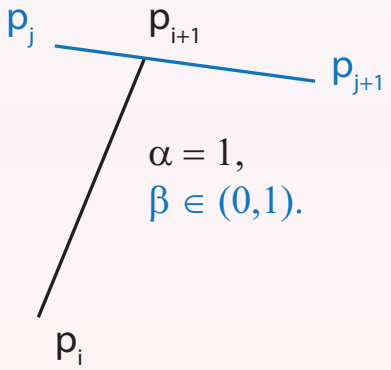


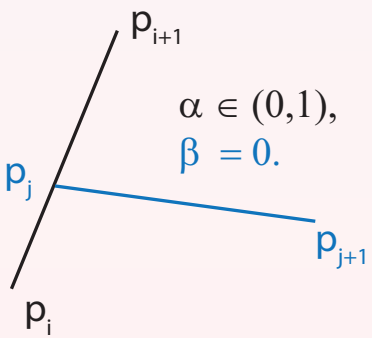


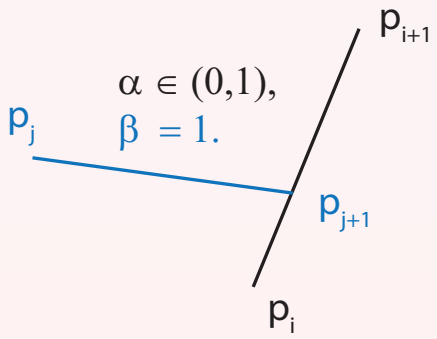


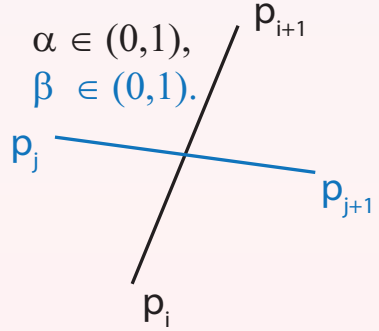












11. Implementační detaily

Odvozená struktura uchovává další vlastnosti:

```
class QPointFB : public QPointF{
   double alpha;                  //Value of α parameter
   double beta;                   //Value of β parameter
   TPointPolygonPosition pos;     //Midpoint p̄ = 0.5(p_i + p_{i+1}) position
```

Kromě parametrů α, β, uchováváme polohu středního bodu hrany jedné oblasti

$$\overline{p}_i = 0.5(p_i + p_{i+1}$$

vůči druhé oblasti; viz 14.

Zpracování průsečíku b:

Oblast P s n vrcholy, i index poč. bodu hrany e, $t \equiv \alpha|\beta$.

Řešeny pouze průsečíky představující vnitřní body e.

Algoritmus 2: ProcessIntersection(b,t,P,i)

1: if ($|t| > \epsilon$ && $||t| - 1| > \epsilon$):

2: $i \leftarrow i + 1$ //Inkrementuj pozici

3: $P \leftarrow (b, i)$ //Přidej průsečík na pozici $i + 1$.

---

12. Výpočet průsečíků, naivní algoritmus

Hledání všech průsečíků b_{ij}.

Pro segment $e_i = (p_i, p_{i+1})$ procházíme postupně všechny segmenty $e'_j = (q_j, q_{j+1})$.

Zásady:

1) Segment e'_j protíná e_i nejvýše jednou

Průsečík b_{ij} lze přidat mezi body q_j a q_{j+1} přímo.

2) Segment e_i může být protnut více segmenty e'_j

Průsečík b_{ij} zatím nelze přidat mezi p_i a p_{i+1}.

Nutno spočítat všechny b_{ij}: e_i testován vůči všem e'_j

$$b_{ij} = e_i \cap e'_j, \qquad i = const, j = 1, ..., m-1.$$

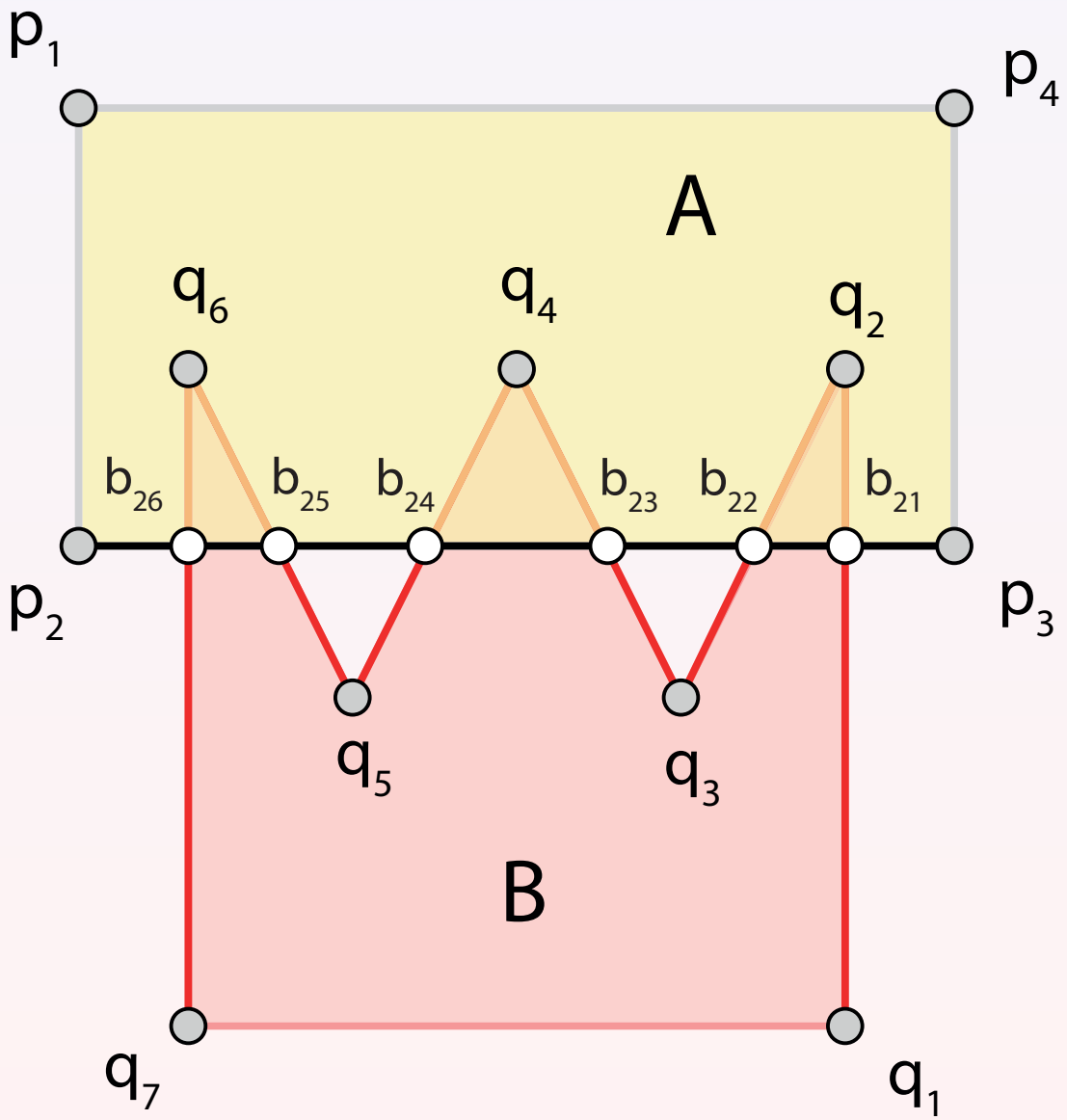
Dvojice (*key*, *val*) uchovávány v mapě M dle α

$$M(\alpha) = (\alpha_i, b_{ij}).$$

Následně M přidána mezi body p_i a p_{i+1}.

Složitost $O(m, n)$.

13. Ukázka průsečíků segmentů



14. Update seznamů A, B

S každým nalezeným průsečíkem $e_i \cap e'_j$, $j = 1, ..., m-1$, updatován B.

Průsečík b_{ij} vložen mezi body p_j, p_{j+1}.

Pro A průsečíky b_{ij} ukládány do $M(\alpha)$.

Po nalezení všech b_{ij} průsečíků $e_i \cap e'_j$ hromadně updatován A.

$M(\alpha)$ přidána mezi body p_i, p_{i+1}.

A		p_1	p_2	p_3	p_4								
B		q_1	q_2	q_3	q_4	q_5	q_6	q_7						
1.	B	q_1	b_{21}	q_2	q_3	q_4	q_5	q_6	q_7					
2.	B	q_1	b_{21}	q_2	b_{22}	q_3	q_4	q_5	q_6	q_7				
3.	B	q_1	b_{21}	q_2	b_{22}	q_3	b_{23}	q_4	q_5	q_6	q_7			
4.	B	q_1	b_{21}	q_2	b_{22}	q_3	b_{23}	q_4	b_{24}	q_5	q_6	q_7		
5.	B	q_1	b_{21}	q_2	b_{22}	q_3	b_{23}	q_4	b_{24}	q_5	b_{25}	q_6	q_7	
6.	B	q_1	b_{21}	q_2	b_{22}	q_3	b_{23}	q_4	b_{24}	q_5	b_{25}	q_6	b_{26}	q_7
7.	A	p_1	p_2	b_{26}	b_{25}	b_{24}	b_{23}	b_{22}	b_{21}	p_3	p_4			

Op.
$\rightarrow$
$\rightarrow$
$\rightarrow$
$\rightarrow$
$\rightarrow$
$\rightarrow$
$\leftarrow$

$M(\alpha)$					
b_{21}					
b_{22}	b_{21}				
b_{23}	b_{22}	b_{21}			
b_{24}	b_{23}	b_{22}	b_{21}		
b_{25}	b_{24}	b_{23}	b_{22}	b_{21}	
b_{26}	b_{25}	b_{24}	b_{23}	b_{22}	b_{21}
b_{26}	b_{25}	b_{24}	b_{23}	b_{22}	b_{21}

15. Algoritmus výpočtu průsečíků

Pro segment $e_i = (p_i, p_{i+1})$ hledány průsečíky b_{ij} se všemi $e'_j = (q_j, q_{j+1})$.
Pokud $\exists b_{ij}$, pro e'_j zpracován přímo (přidán nebo updatován q_j / q_{j+1}).
Pro segment e_i přidán do M (zatím neznáme všechny b_{ij}).
Po nalezení všech b_{ij} postupně sekvenčně přidány z M.

Algoritmus 2: ComputeIntersections(A, B)

1: for ($i = 0$; $i < n$;i++):
2: $M = map\langle double, QPointFB\rangle$ //Vytvoření mapy
3: for ($j = 0$; $j < m$;j++):
4: if $b_{ij} = (p_i, p_{(i+1)\%m}) \cap (q_j, q_{(j+1)\%m}) \neq \emptyset$ //Existuje průsečík
5: $M[\alpha_i] \leftarrow b_{ij}$ //Přidej do M
6: ProcessIntersection(b_{ij},β,B,j) //Zpracuj první průsečík pro e'_j
7: if ($\|M\| > 0$) //Nějaké průsečíky jsme nalezli
8: for $\forall m \in M$: //Procházej všechny průsečíky v M
9: $b \leftarrow m.second$ //Získej 2. hodnotu z páru
10: ProcessIntersection(b,α,A,i) //Zpracuj aktuální průsečík pro e_i

---

16. Poloha hran A, B

O poloze $e_i = (p_i, p_{i+1})$ k B rozhoduje poloha libovolného vnitřního bodu

$$\overline{p}_i = 0.5(p_i + p_{i+1}),$$

analogicky pro $e'_j = (q_j, q_{j+1})$ vzhledem k A

$$\overline{q}_j = 0.5(q_j + q_{j+1}).$$

Každému $\overline{p}_i \in A$ určíme polohu vzhledem k B.

Každému $\overline{q}_j \in B$ určíme polohu vzhledem k A.

Ohodnocovací fce g

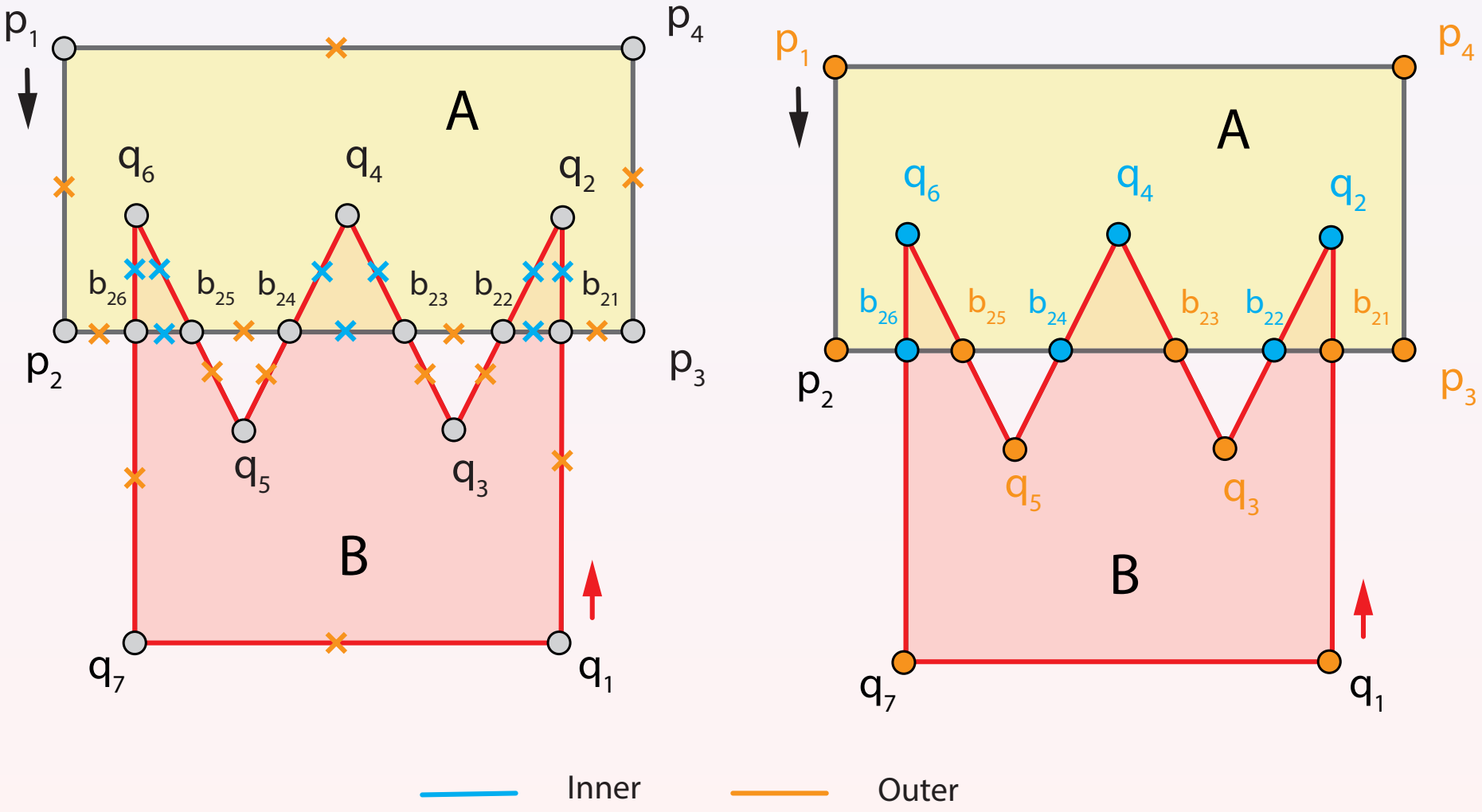
$$g(e_i, B) = g(\overline{p}_i, B) = \begin{cases} OUT, & \overline{p}_i \notin B, \\ BND, & \overline{p}_i \in \partial B, \\ IN, & \overline{p}_i \in B, \end{cases} \qquad g(e'_j, A) = g\overline{q}_j, A) = \begin{cases} OUT, & \overline{q}_j \notin A, \\ BND, & \overline{q}_j \in \partial A, \\ IN, & \overline{q}_j \in A. \end{cases}$$

Hrany rozděleny na vnitřní, vnější, ležící na hranci A/B.

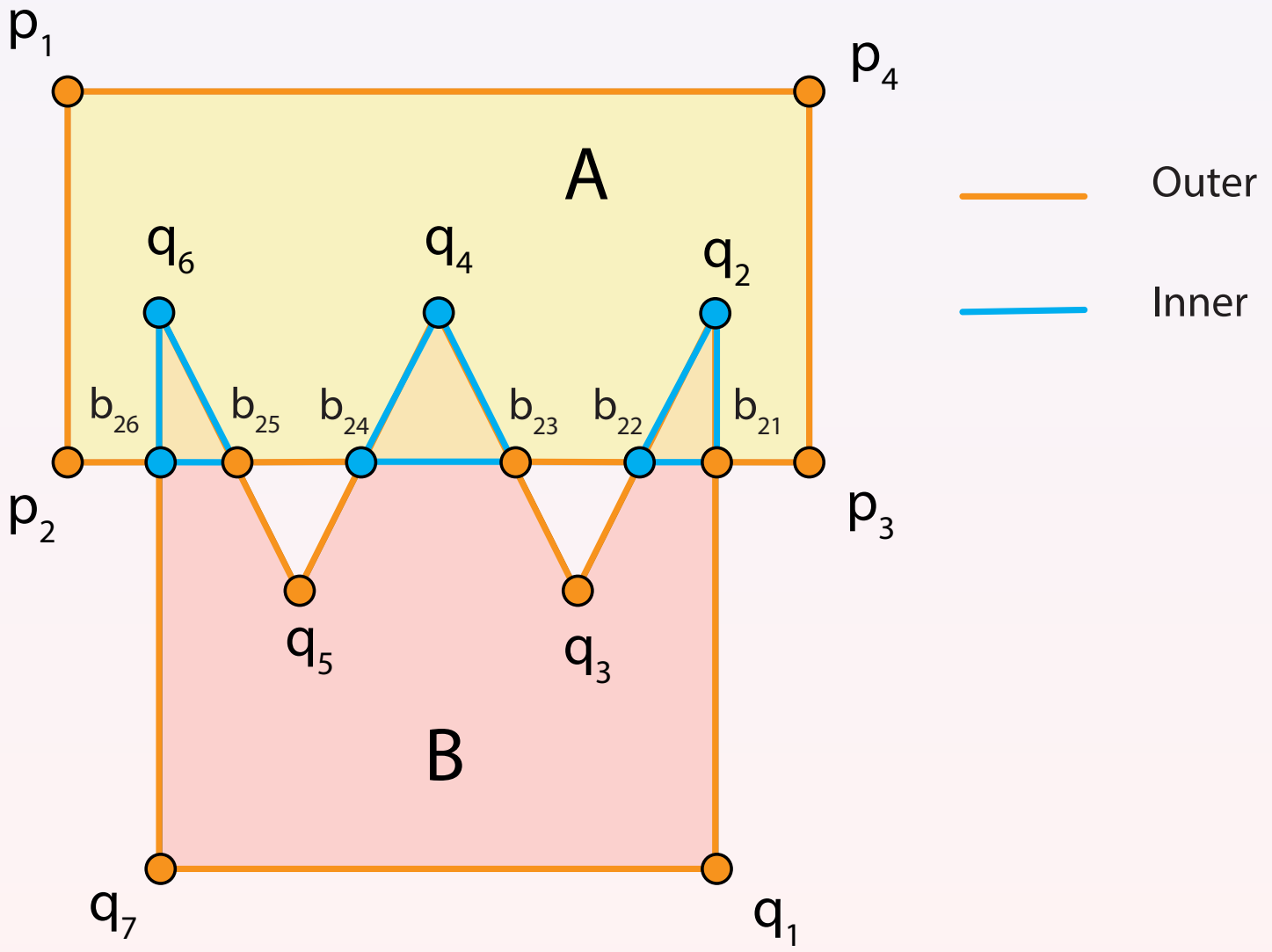
Tato informace uložena do počátečního bodu hrany :

`TPointPolygonPosition pos;` `//Midpoint` $\overline{p} = 0.5(p_i + p_{i+1})$ `position`

17. Ukázka ohodnocení vrcholů *A*, *B*



18. Ukázka ohodnocení hran *A*, *B*



19. Ukázka ohodnocení hran A, B

Ohodnocení hran vrcholů A, B:

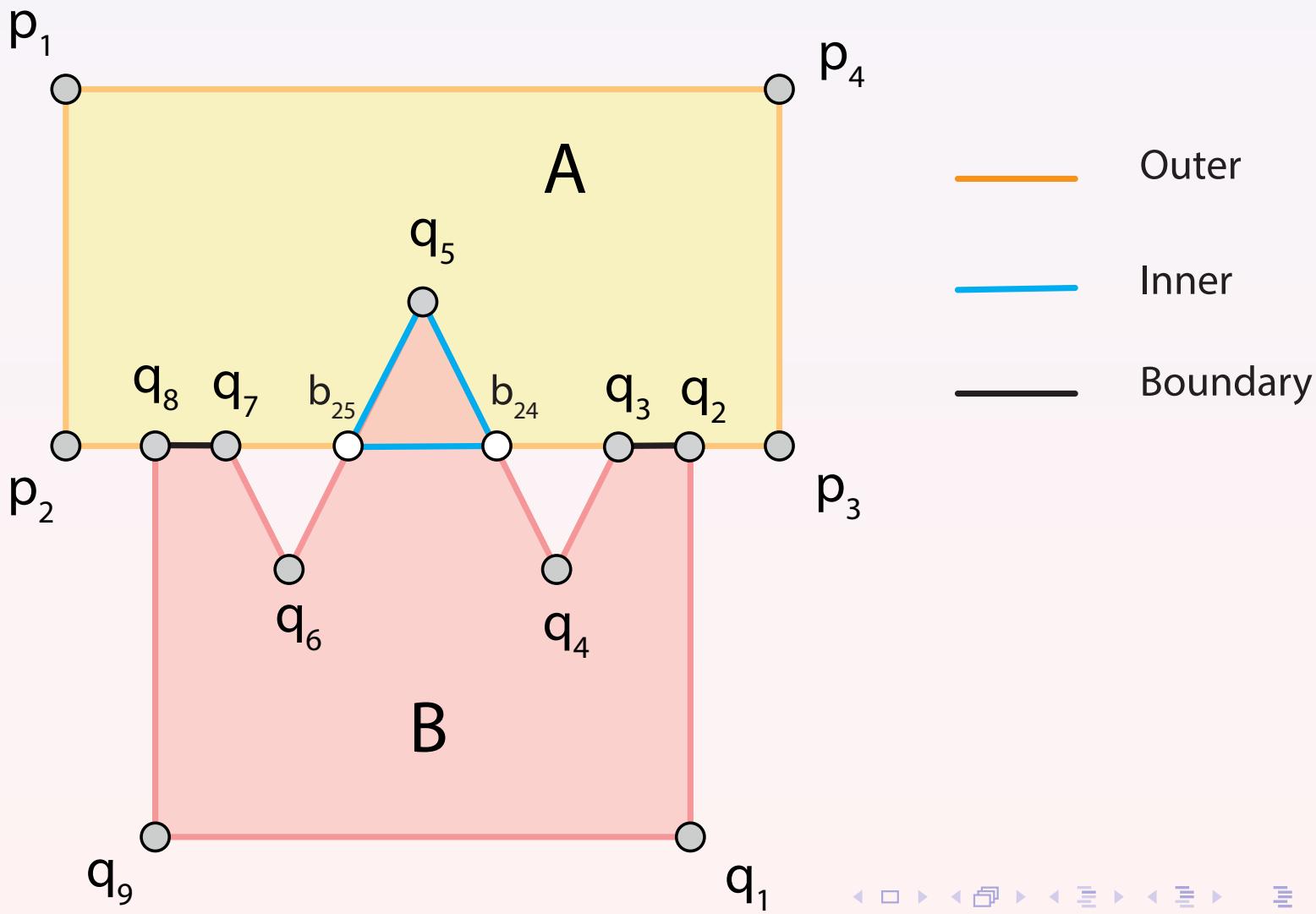
A	p_1	p_2	b_{26}	b_{25}	b_{24}	b_{23}	b_{22}	b_{21}	p_3	p_4			
$g(e_i, B)$	O	O	I	O	I	O	I	O	O	O			
B	q_1	b_{21}	q_2	b_{22}	q_3	b_{23}	q_4	b_{24}	q_5	b_{25}	q_6	b_{26}	q_7
$g(e'_j, A)$	O	I	I	O	O	I	I	O	O	I	I	O	O

V A, B se nevyskytují singulární segmenty (kolineární).

Chybí ohodnocení BND (boundary).

20. Singulární *A*, *B*

Přibude ohodnocení BND, kolineární segmenty.



21. Výběr hran dle ohodnocení

Výběr hran A, B dle prováděné množinové operace.

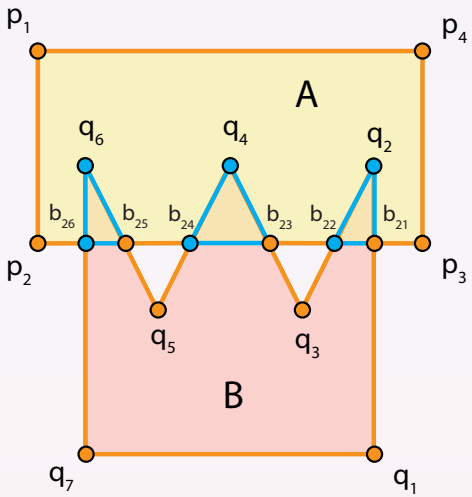
Z hran následně sestaveny fragmenty.

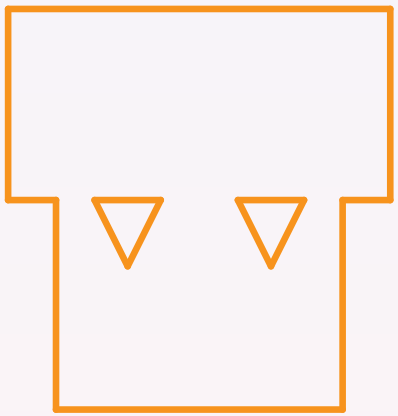
Operace	A	B
$C = A \cap B$	Inner	Inner
$C = A \cup B$	Outer	Outer
$C = A \cap \overline{B}$	Outer	Inner
$C = B \cap \overline{A}$	Inner	Outer
$C = A \triangle B$,	Inner + Outer	Inner + Outer

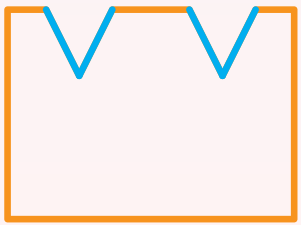
Nestabilita: nepřesné určení polohy bodu a oblasti.

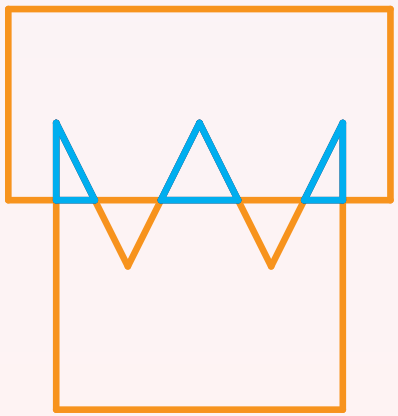
Pro singulární případy atribut BND.

22. Výsledek množinových operací



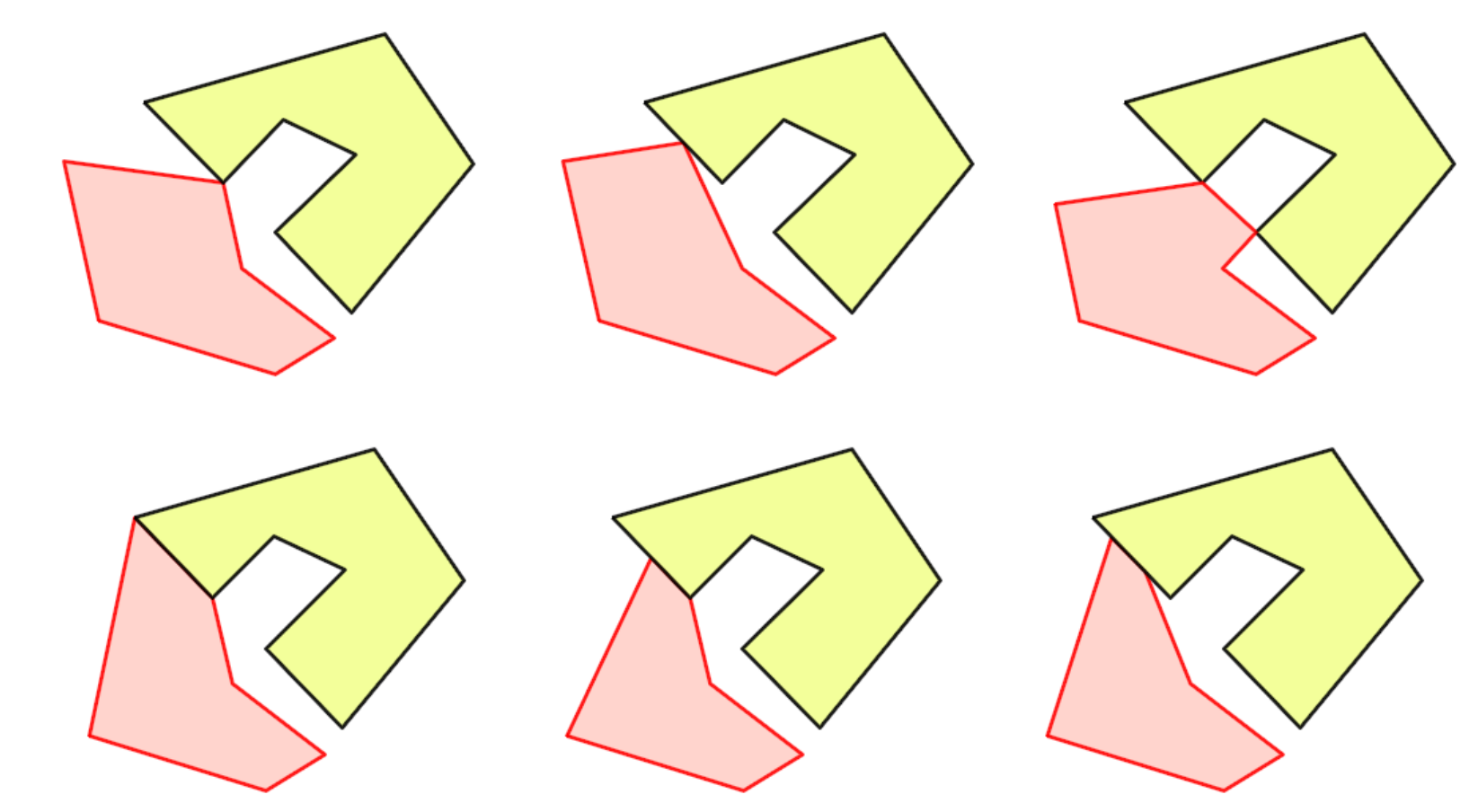






23. Problematické situace

Při odladění algoritmu nutné zohlednit řadu singularit: např. oblasti mající společný pouze jeden vrchol, stranu, část strany...



24. Formulace problému

Dáno:	Uzavřená a ohraničená oblast C v $\mathbb{R}^2$či linie.
Hledáme:	Offset C_d oblasti C.

Definice 1:

Offset C_d oblasti C představuje ekvidistantu oblasti tvořenou hraniční křivkou této oblasti $c(t)$ ve vzdálenosti d.

Offset obecné oblasti nemusí být racionální křivkou.

Rovnoběžné křivky (tzv. Bertrandovy křivky) mají společné hlavní normály (platí pro C a C_d).

Definice 2:

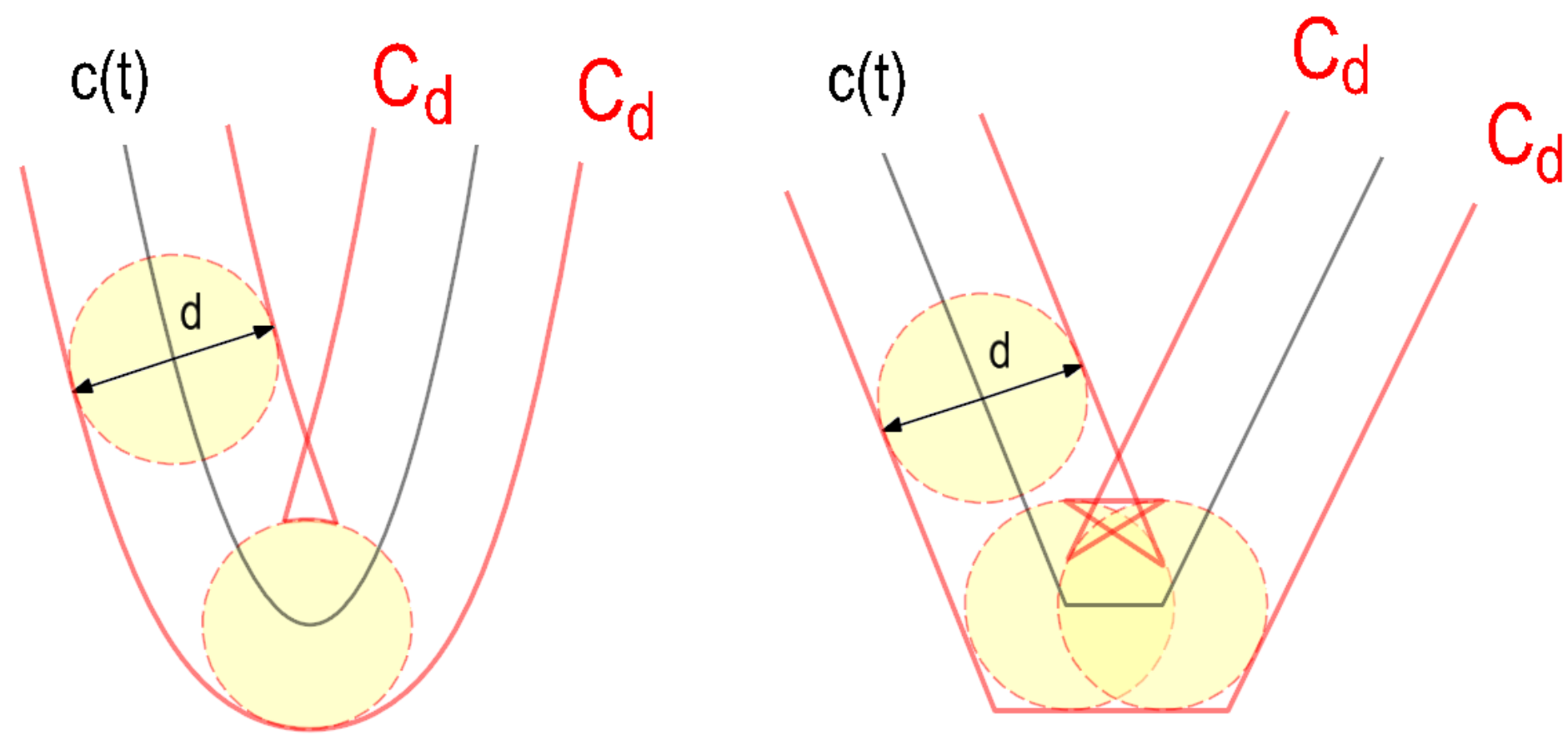
Offset C_d definován jako obálka soustavy kružnic s poloměrem d, jejichž střed se pohybuje po $c(t)$.

Offset C_d může protínat sám sebe, na C_d.

Vznikají singulární body, jejichž vzdálenost od C je menší než d $\Rightarrow$singulární část offsetu.

25. Ukázka offsetu C_d

Offset C^d obsahující singulární body.



26. Konstrukce offsetu

Metody konstrukce offsetu:

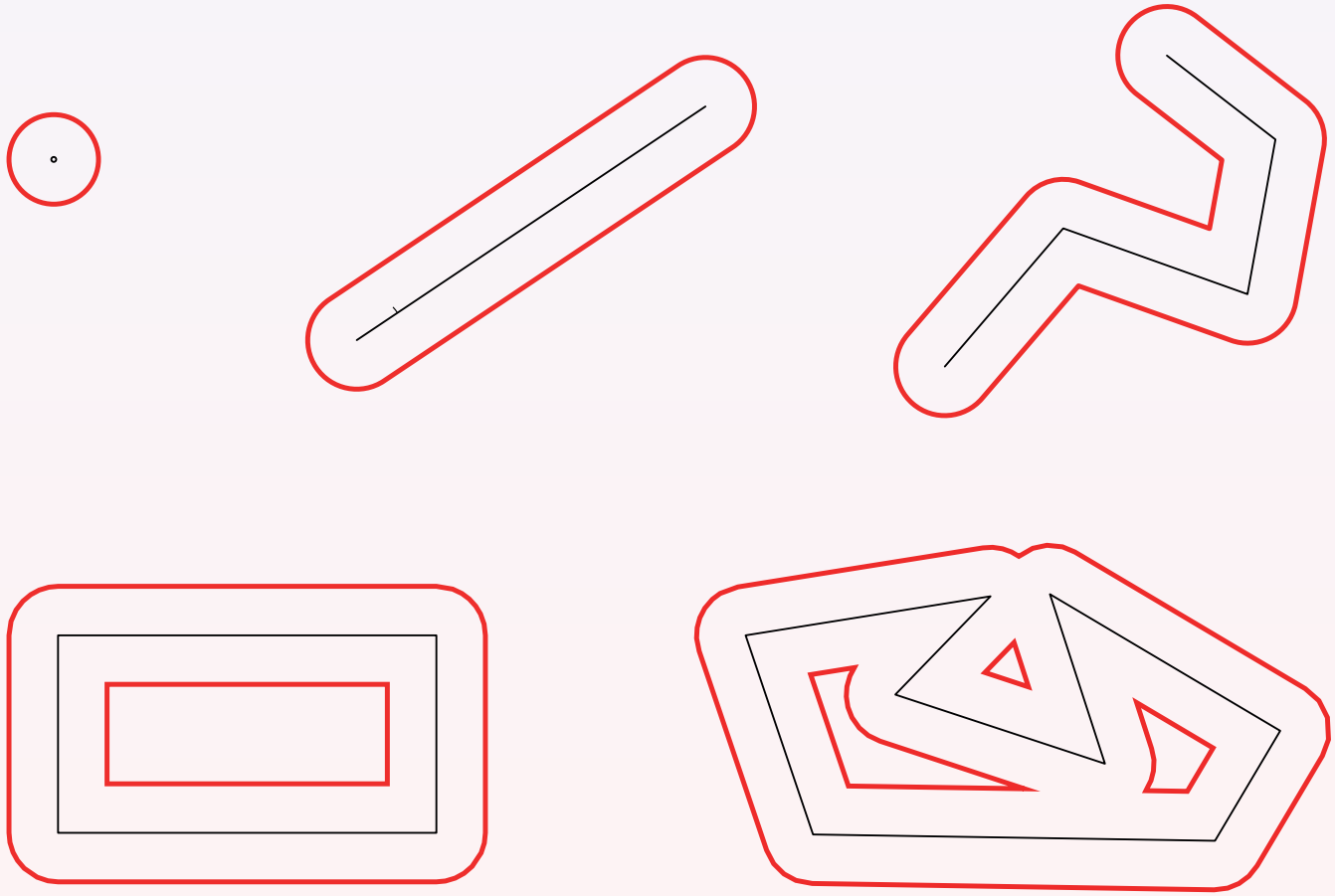
1. *Analytická konstrukce*
   Použití pro linie, konvexní oblasti, snadná implementace.
   Oblouky vzorkovány (náhrada konvexním $n-$úhelníkem).
2. *Využití Winding number*
   Offset segmentů a odstranění smyček aplikací WA.
3. *Přibližná konstrukce*
   Straight skeleton, místo smršťování roztažení.
   Netřeba provádět dekompozici.
   Chybí zaoblené segmenty (kružnicové oblouky).
4. *Minkowského suma*
   Nekonvexní oblasti: dekompozice na konvexní (triangulace).
   Nad každou oblast Minkowského suma + dissolving.
   Alternativně lze použít konvoluci.

Topologicky korektní implementace je náročná.

Obecné problémy: numerická přesnost, stabilita, množství singularit.

Dissolving: opakované sjednocení bufferů nad jednotlivými segmenty.

27. Ukázky offsetu pro různé prvky

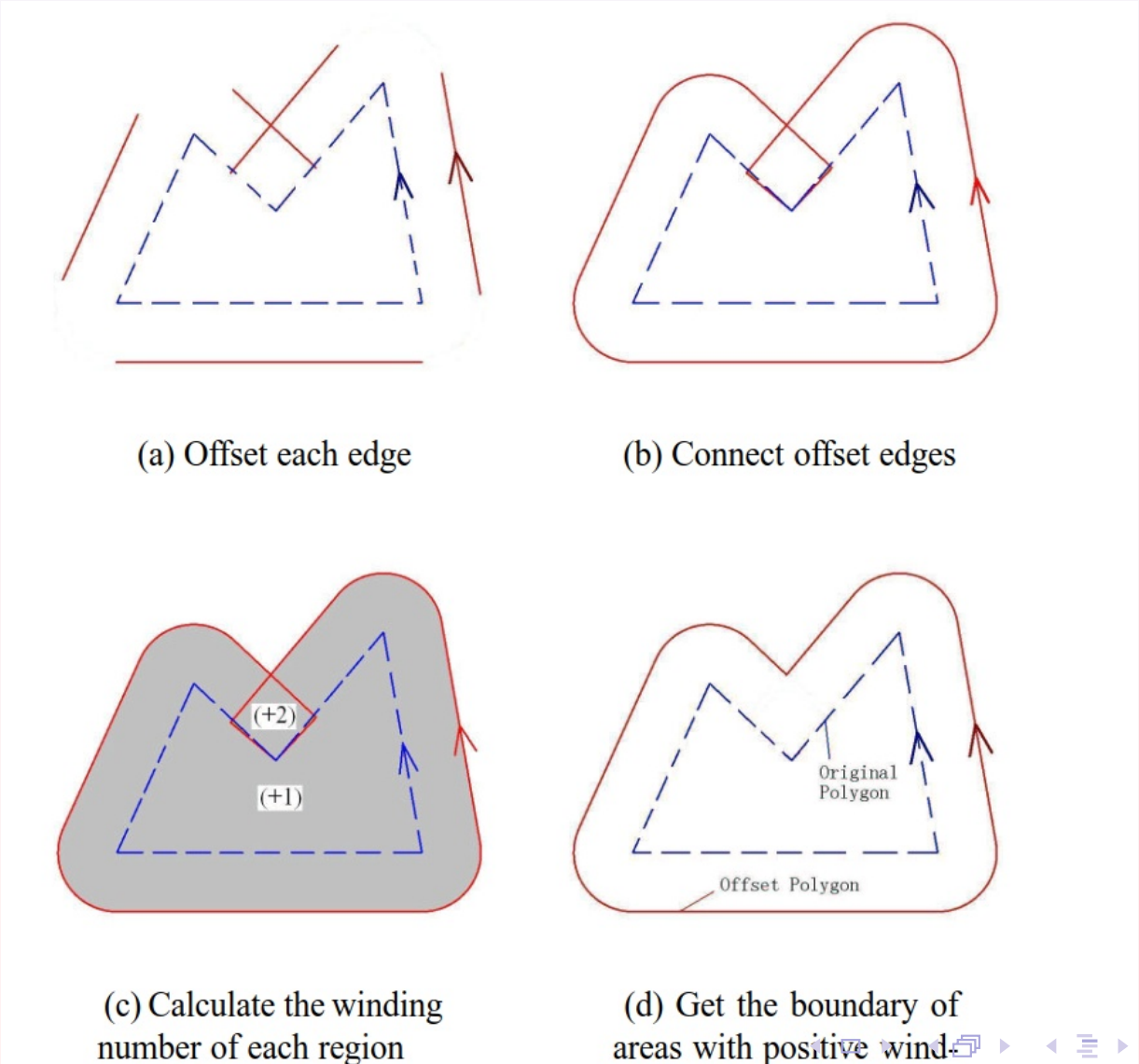


28. Ukázky offsetu s využitím WA

Posun každého segmentu

Propojení s původními vrcholy $\Rightarrow$ non-simple polygon.

Non-simple části polygonu odstraněny WA.



(a) Offset each edge

(b) Connect offset edges

(c) Calculate the winding number of each region

(d) Get the boundary of areas with positive wind-

29. Exaktní konstrukce offsetu, liniové segmenty

Definice offsetu nad segmentem $e_i = (p_i, p_{i+1})$

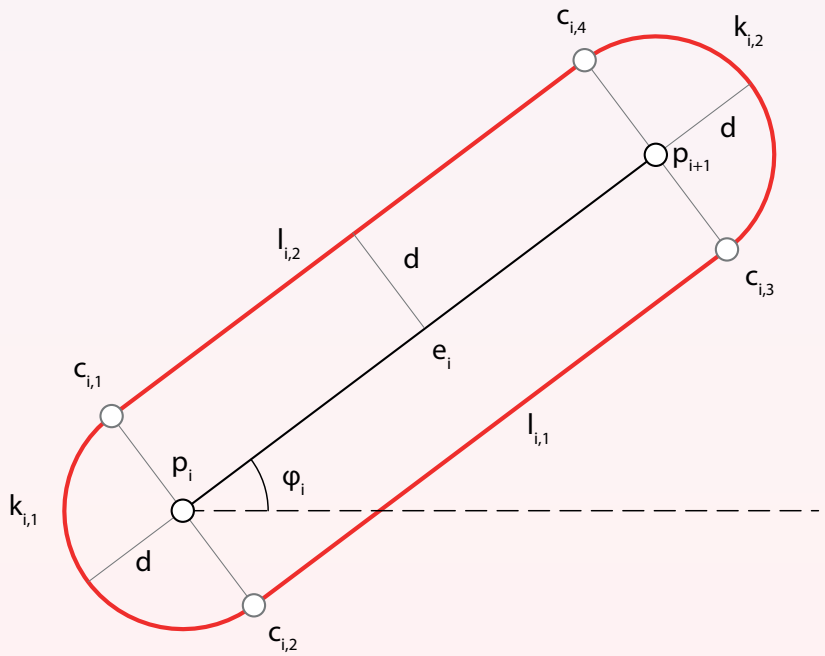
$$C_d(e_i) = \{c \mid \|c - e_i\| = d\}.$$

Tvořen 2 úsečkami a 2 kružnicovými oblouky

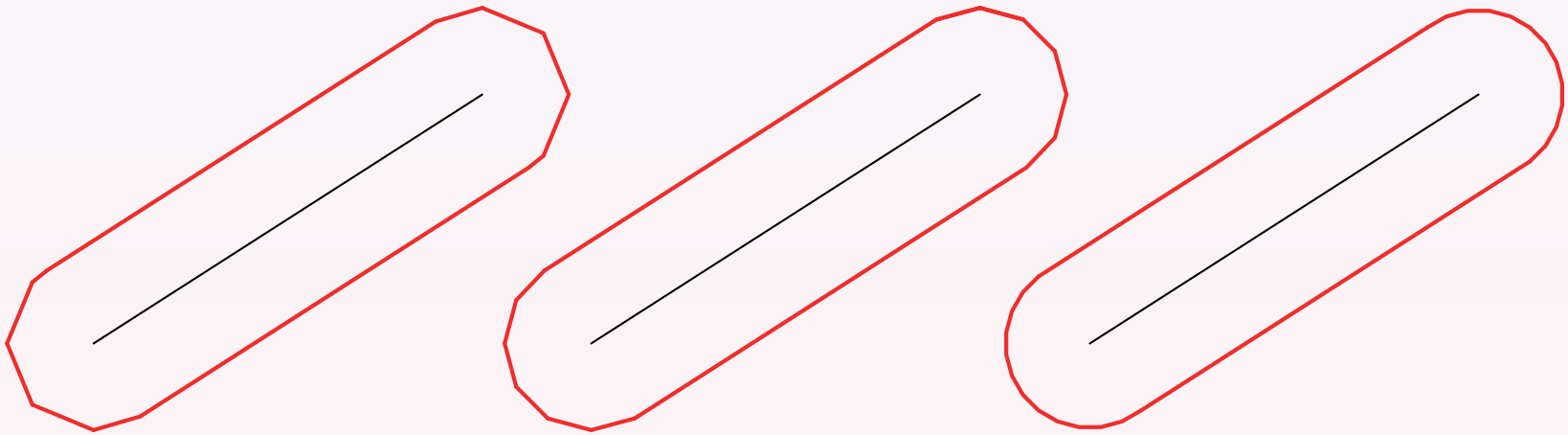
$$C_d(e_i) = \left\{k_{i,1}, l_{i,1}, k_{i,2}, l_{i,2} \mid \left\|k_{i,1} - p_{i+1}\right\| = \left\|l_{i,1} - e_i\right\| = \left\|k_{i,2} - p_i\right\| = \left\|l_{i,2} - e_i\right\| = d\right\},$$

kde

$$k_{i,1}(c_{i,1}, c_{i,2}, d), l_{i,1} = (c_{i,2}, c_{i,3}), k_{i,2}(c_{i,3}, c_{i,4}, d), l_{i,2} = (c_{i,4}, c_{i,1}).$$



30. Liniový offset, různý krok vzorkování



29. Offset, polylinie

Polylinie $L = \{p_1, ..., p_n\}$.

Nad každým segmentem $e_i = (p_i, p_{i+1})$ vygenerován liniový buffer

$$C_d(e_i) = \{c \mid \|c - e_i\| = d\}.$$

Dissolving:

Sjednocení n liniových bufferů nad L

$$C_d(L_n) = \bigcup_{i=1}^{n} C_d(e_i).$$

Reflexní vrchol :

Generuje kružnicový oblouk, jinak liniové segmenty.

Může vést k rozpadu $C_d(L)$ na vnitřní a vnější offset

$$C_d(L) = \{C_{d,o}(L), C_{d,i}(L) \mid C_{d,i}(L) \in C_{d,o}(L)\}.$$

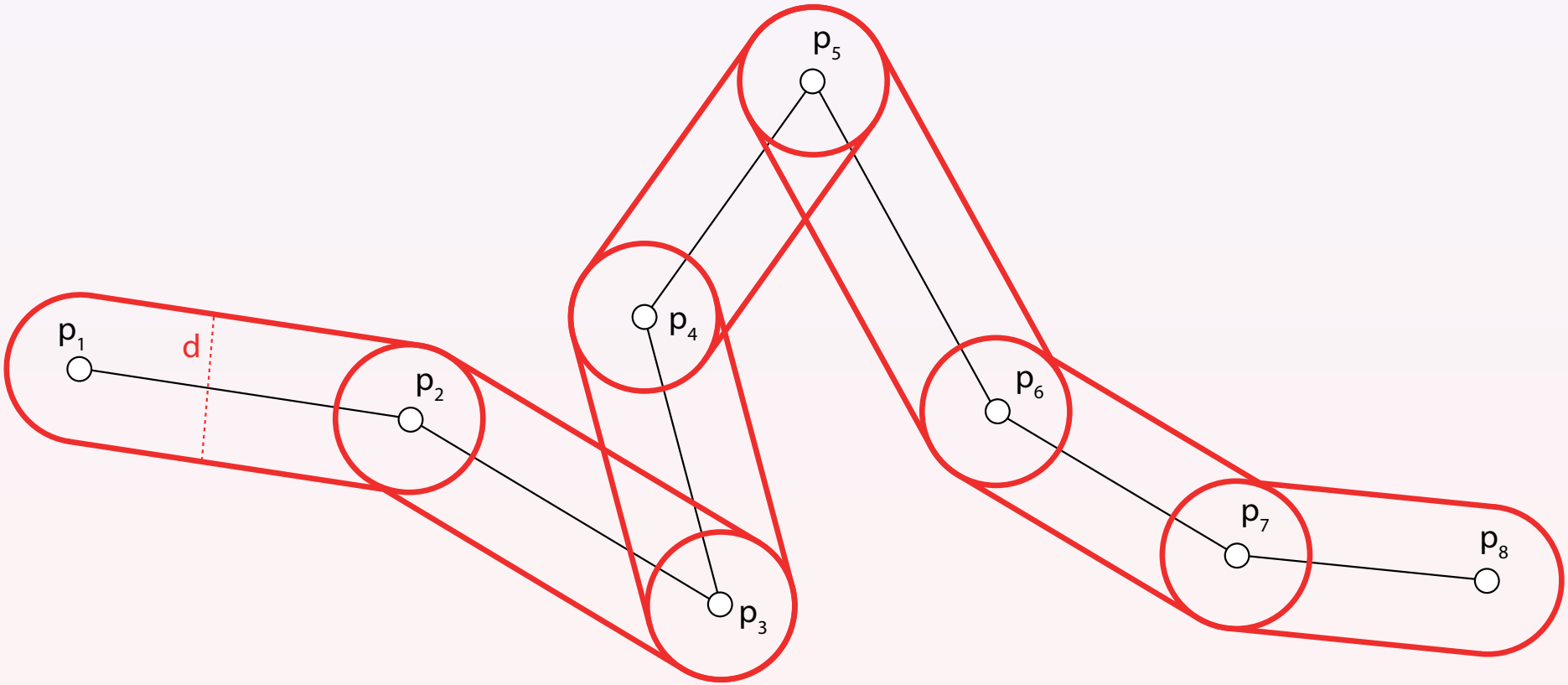
Vnitřní offset nemusí existovat pro různá d.

Inkrementální algoritmus:

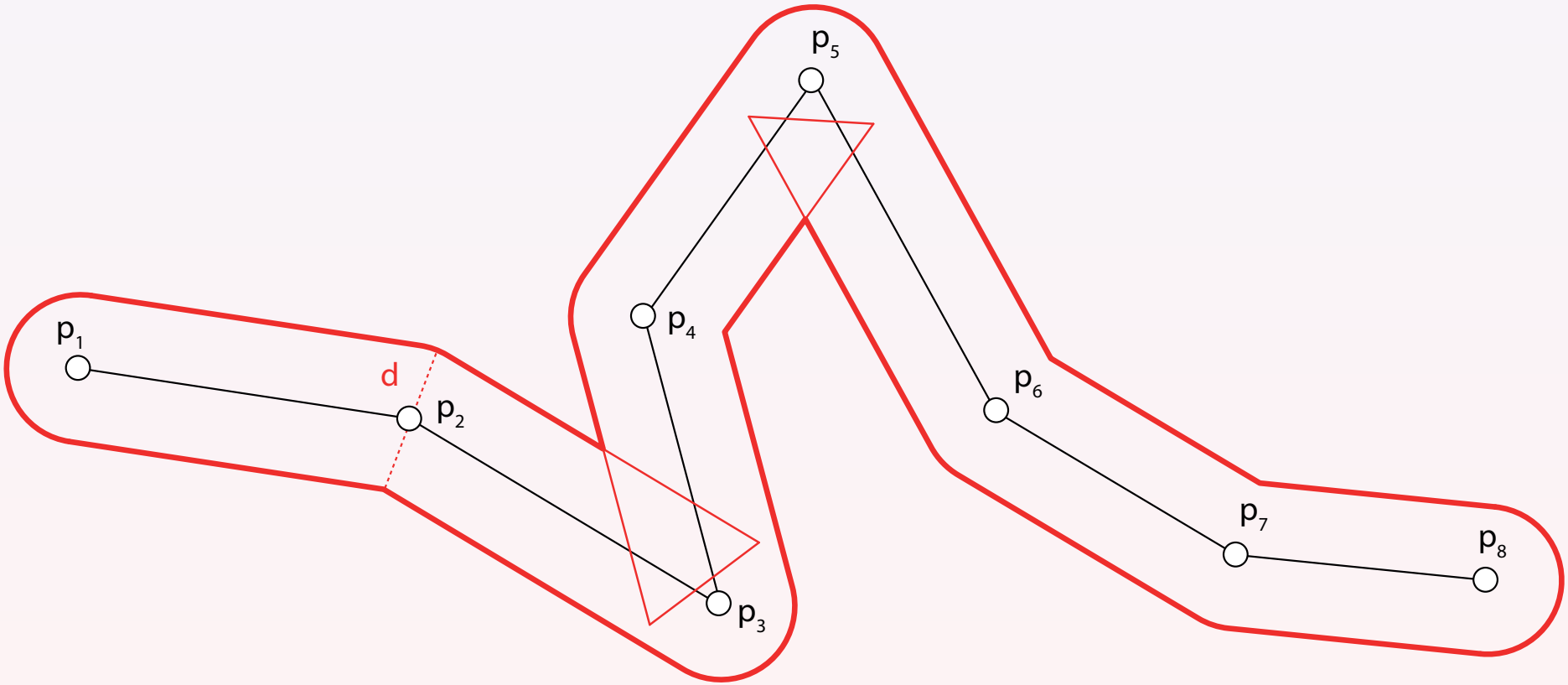
Offset nad k segmenty polylinie L z offsetu pro $k - 1$ segmenty.

Tento následně sjednocen s offsetem k-tého segmentu

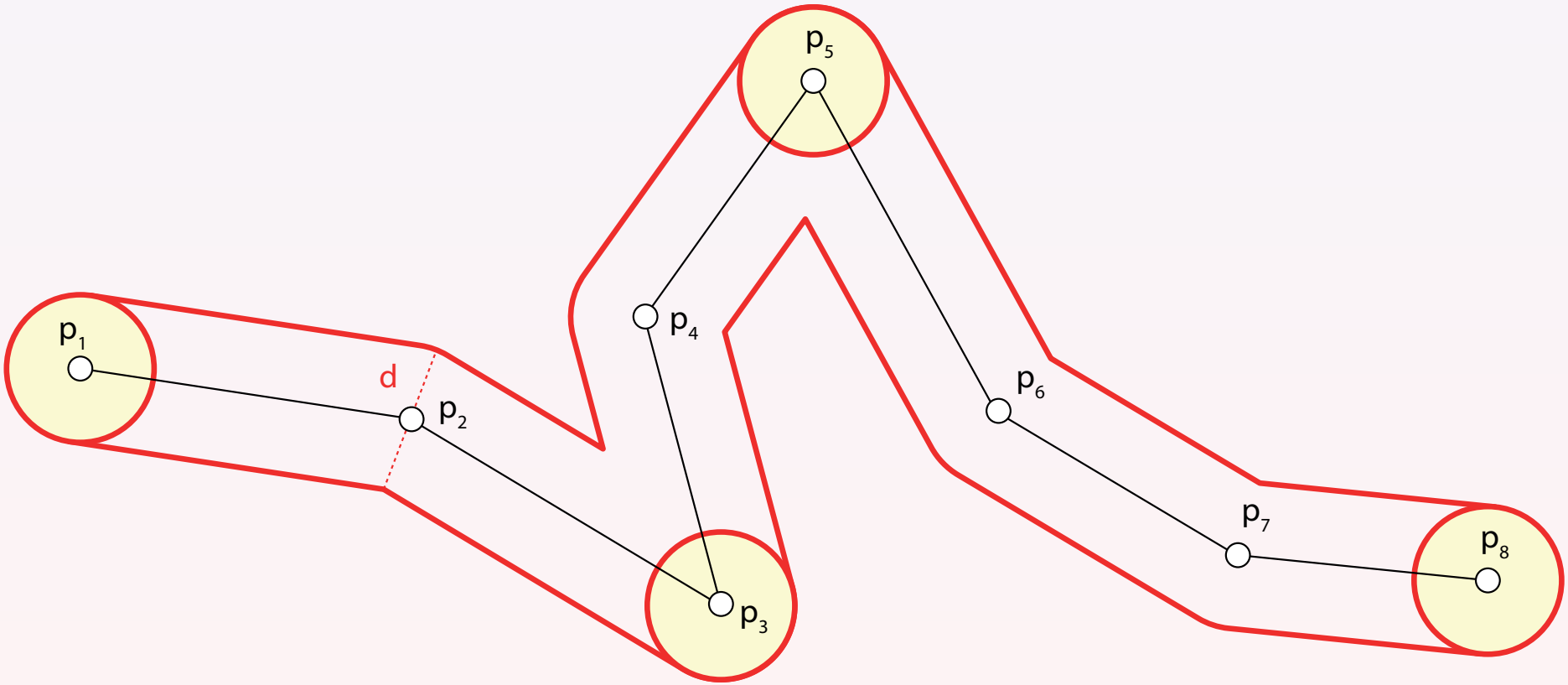
30. Offset nad segmenty polylinie



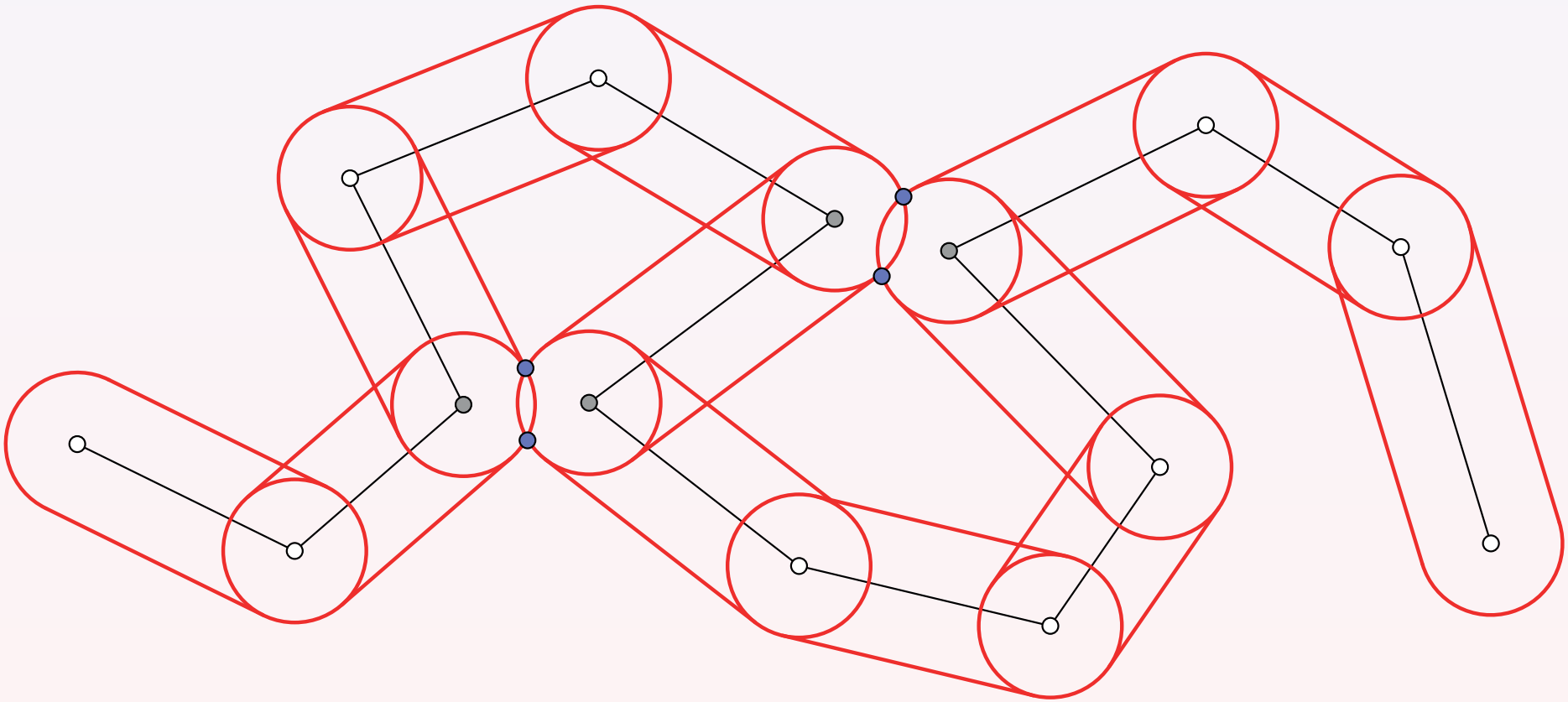
31. Offset polylinie: dissolve, singularity



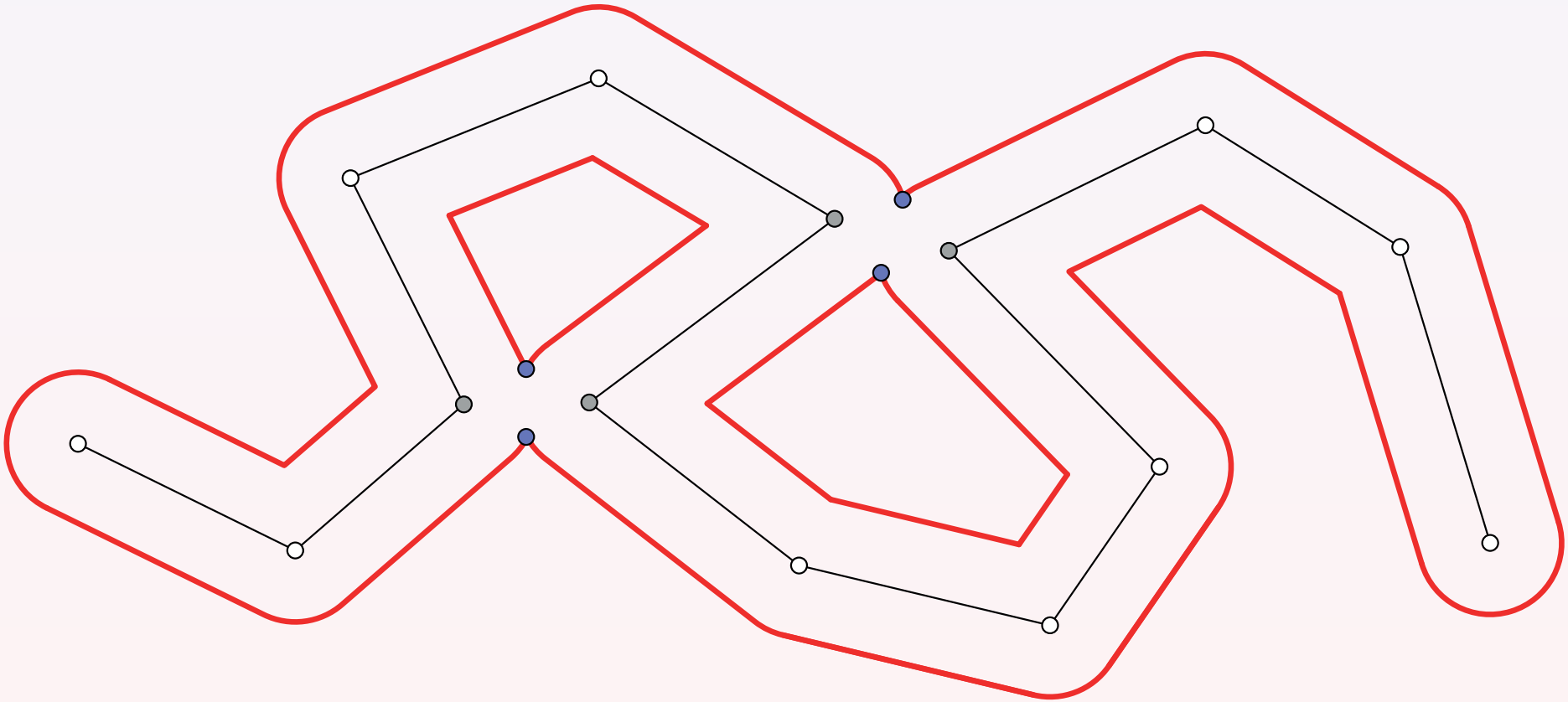
32. Offset polylinie: dissolve, bez singularit (GIS)



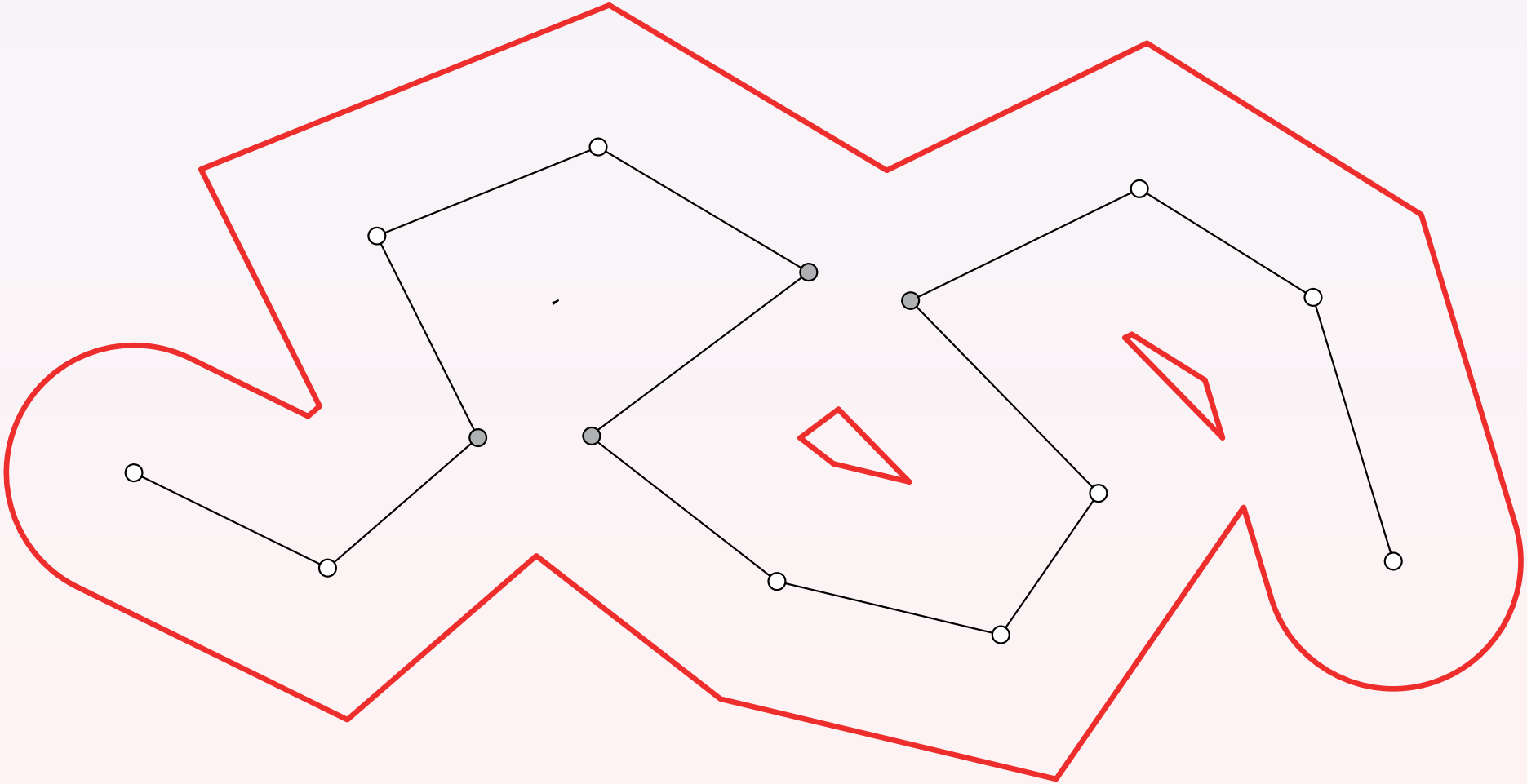
33. Reflexní vrchol, vnitří/vnější offset



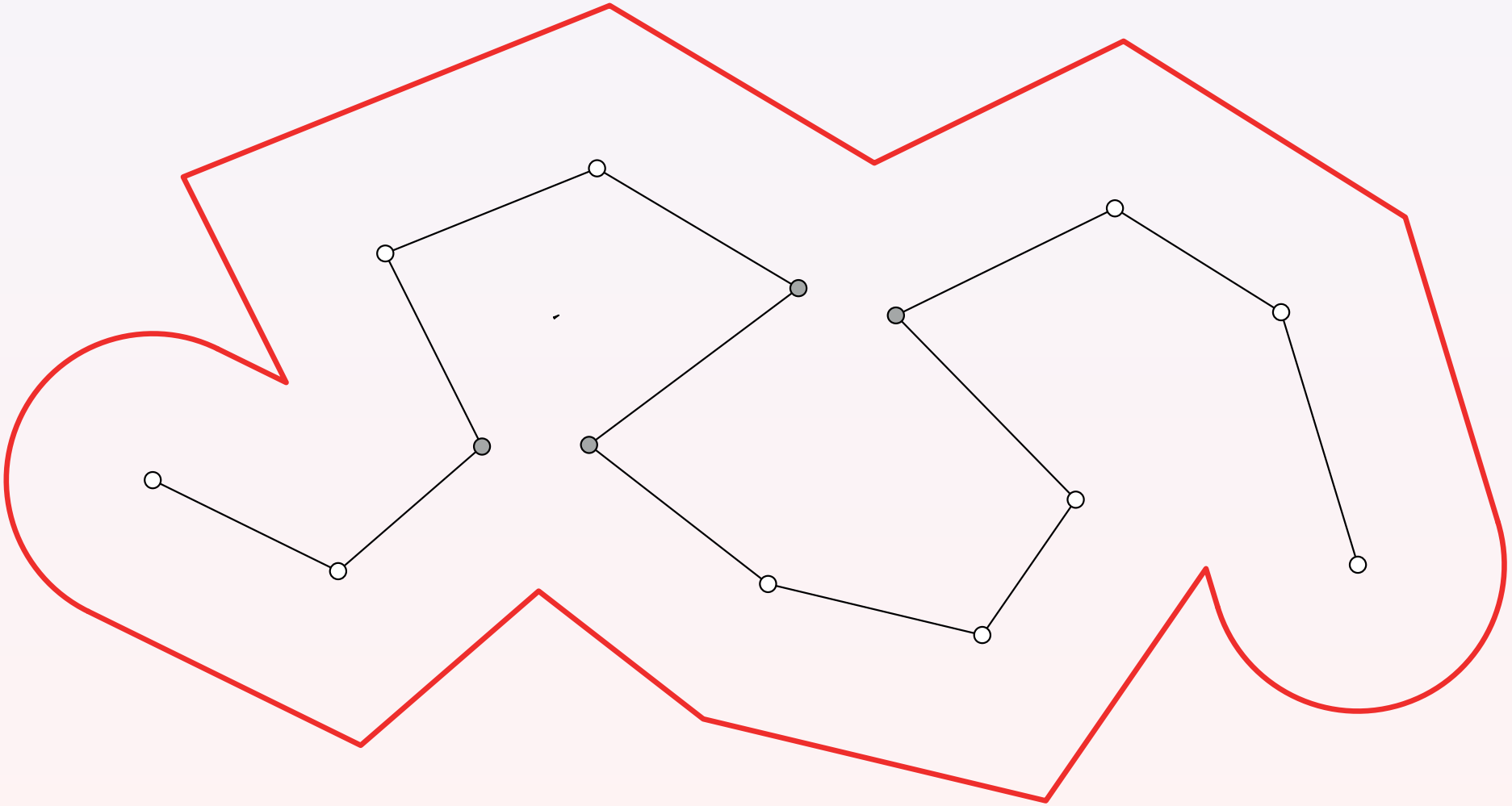
34. Reflexní vrchol, vnitří/vnější offset: dissolving



35. Zánik vnitřního offsetu I.



36. Zánik vnitřního offsetu II.



37. Offset uzavřené oblasti

Uzavřená oblast $P = \{p_1, ..., p_n\}$.

Nad každým segmentem $e_i = (p_i, p_{i+1})$ vygenerován liniový buffer

$$C_d(e_i) = \{c \mid \|c - e_i\| = d\}.$$

Proces dissolvingu

$$C_d(P_n) = \bigcup_{i=1}^{n} C_d(e_i).$$

Reflexní vrchol opět generuje kružnicový oblouk, jinak liniové segmenty.

Může vést k rozpadu na vnitřní a vnější offset.

Vždy existuje vnější offset, v některých případech i vnitřní offset

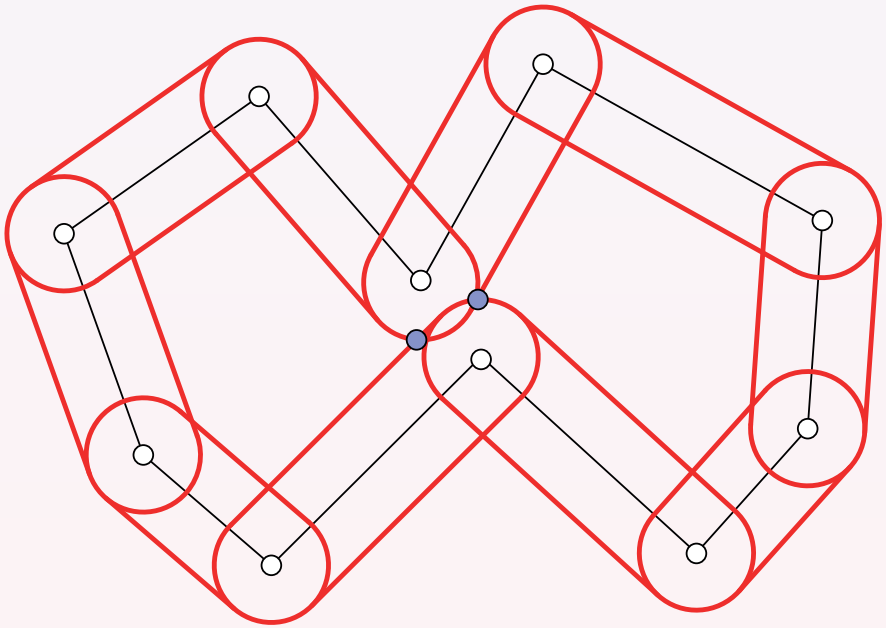
Inkrementální konstrukce:

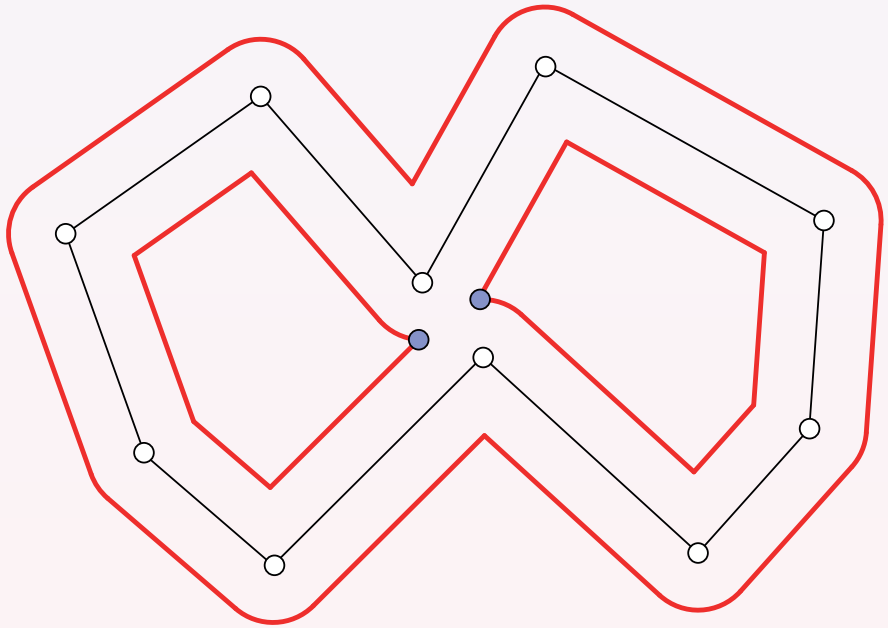
Offset nad k segmenty oblasti P z offsetu pro $k - 1$ segmentů P.

Poté sjednocen s offsetem k-tého segmentu

$$C_d(L_k) = C_d(L_{k-1}) \cup C_d(e_k).$$

38. Rozpad offsetu





39. Minkowského suma v E_2

Definice:

Nechť $\mathcal{A}, \mathcal{B}$ jsou dvě množiny bodů v E_2. Minkowského suma $\mathcal{A} \oplus \mathcal{B}$ je definována jako

$$\mathcal{A} \oplus \mathcal{B} = \bigcup_{b \in B} \mathcal{A}^b,$$

kde množina $\mathcal{A}^b = \{a + b \mid a \in \mathcal{A} \wedge b \in \mathcal{B}\} = \mathcal{A} + b$ je množina $\mathcal{A}$ posunutá o vektor b.

Komutativita a asociativita Minkowského sumy:

$$\mathcal{A} \oplus \mathcal{B} = \mathcal{B} \oplus \mathcal{A},$$
$$(\mathcal{A} \oplus \mathcal{B}) \oplus \mathcal{C} = \mathcal{A} \oplus (\mathcal{B} \oplus \mathcal{C}.$$

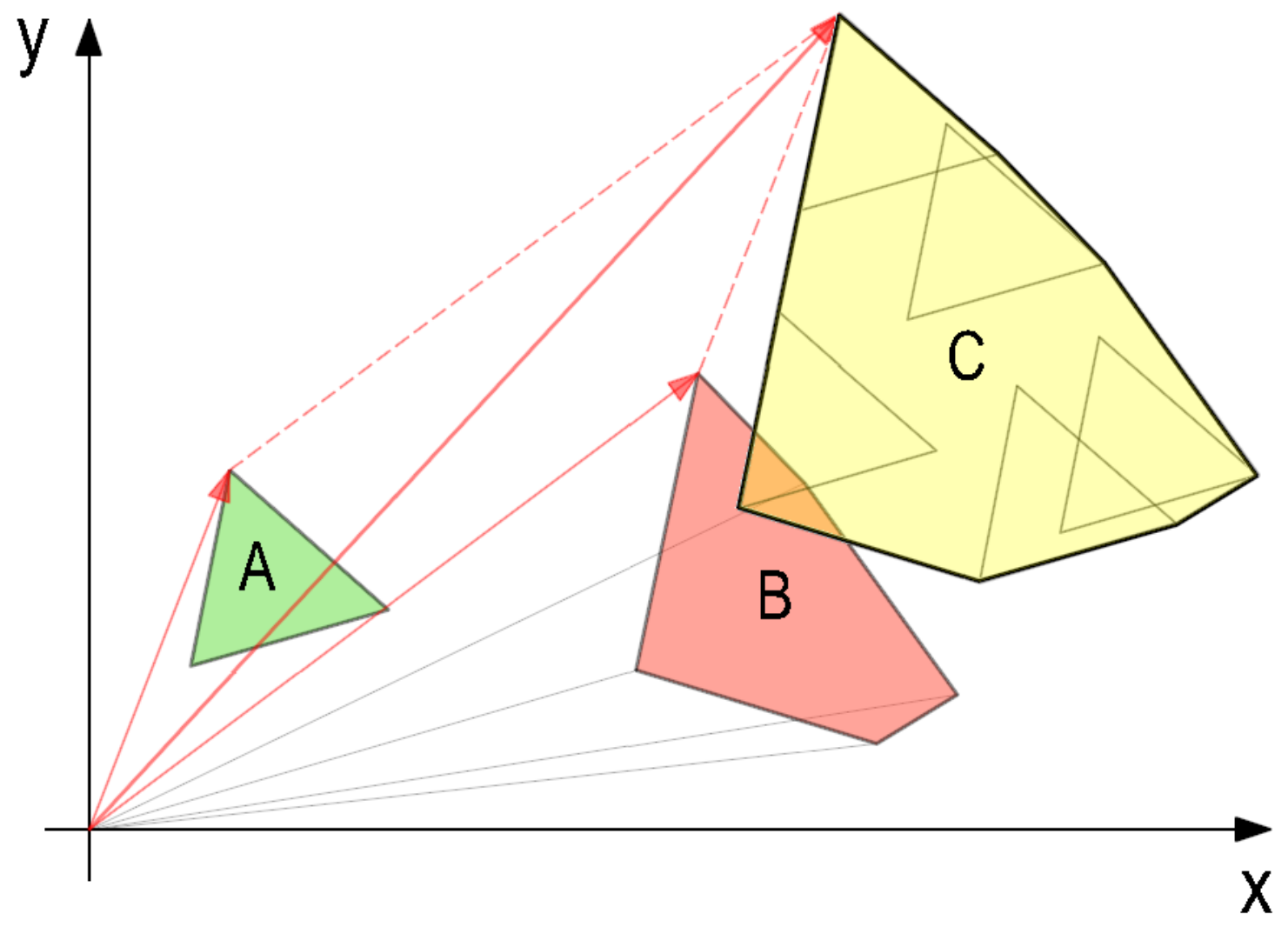
Minkowského suma 2 konvexních resp. nekonvexních množin je konvexní resp. konvexní nebo nekonvexní množina.

Nechť A_i a B_j jsou množiny v $\mathbb{E}_2$, pak

$$\bigcup_i A_i \oplus \bigcup_j B_j = \bigcup_{i,j} (A_i \oplus B_j).$$

Tato vlastnost používána při hledání Minkowského sumy nekonvexních množin.

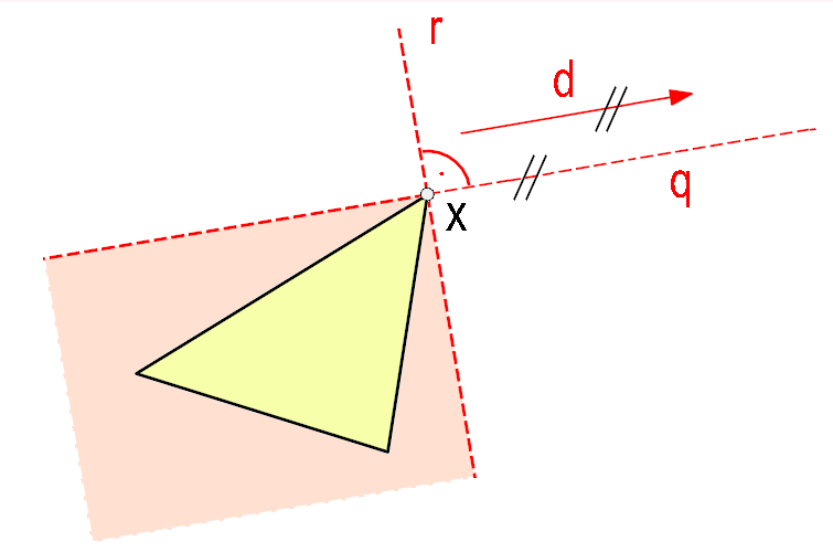
40. Ilustrace Minkowského sumy



41. Extrémní bod

Extrémní bod množiny $\mathcal{A}$ ve směru daném vektorem d:

Takový bod $x \in \mathcal{A}$, pro který všechny ostatní body množiny $\mathcal{A}$ leží v polorovině (q, r), kde $q(x, d)$ a $r(x, \perp d)$.

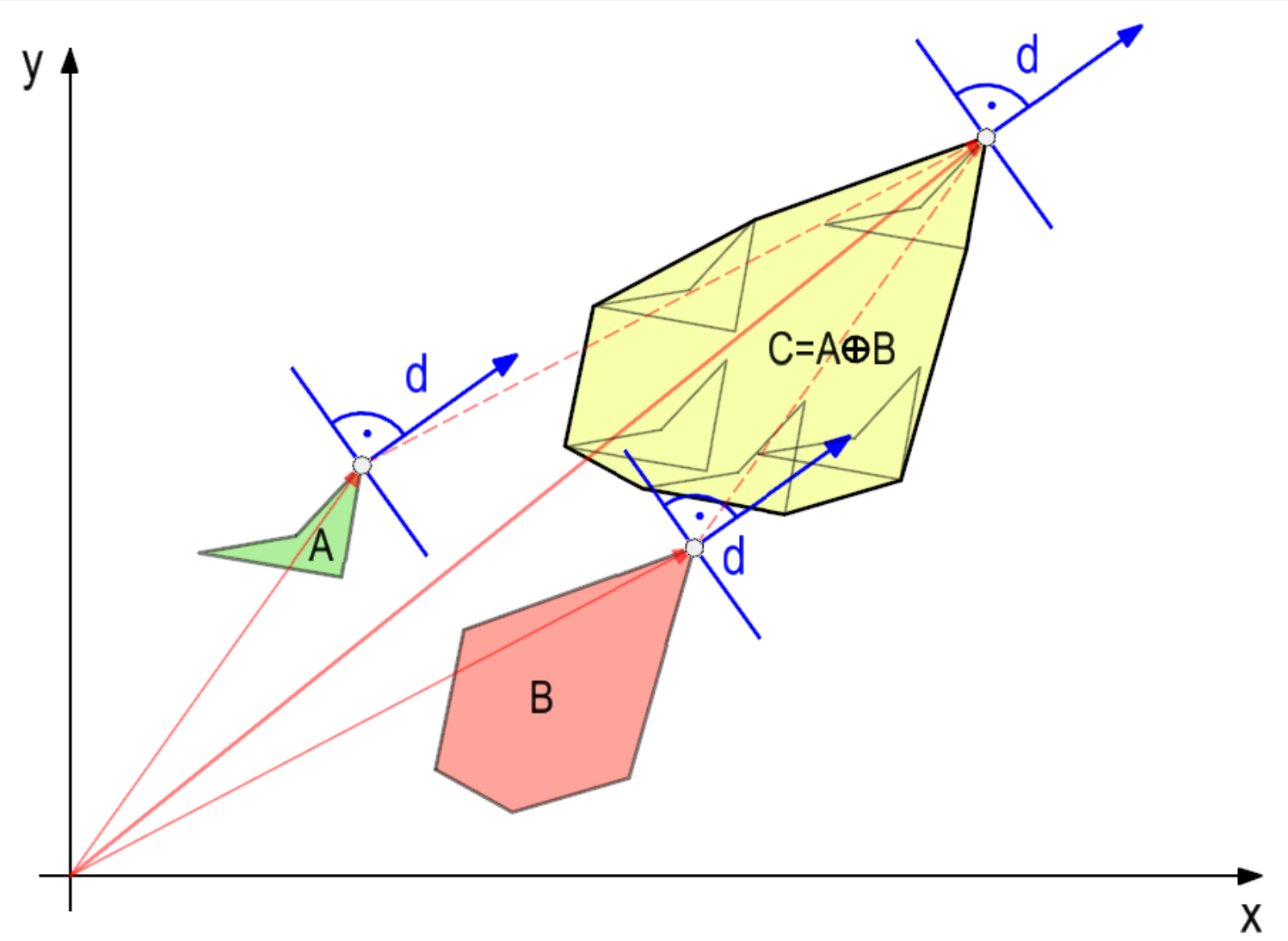


Nalezení extrémního bodu:

Extrémní bod C ve směru d je součtem extrémních bodů $\mathcal{A}, \mathcal{B}$ ve směru d.

Důležité pro konstrukci Minkowského sumy.

42. Znázornění vztahu extrémních bodů

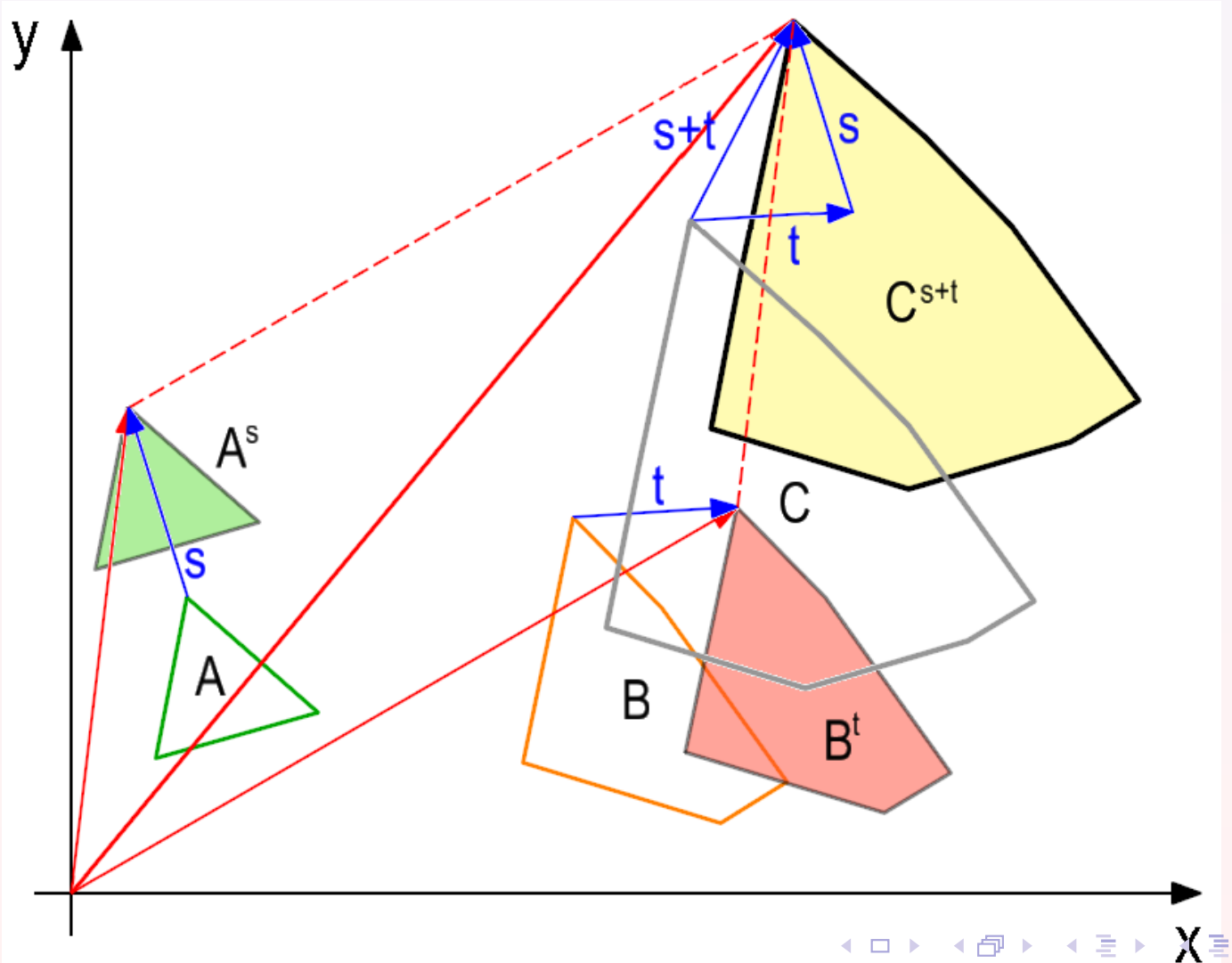


43. Invariance vůči posunu

Posun $\mathcal{A}, \mathcal{B}$ o libovolné vektory s, t

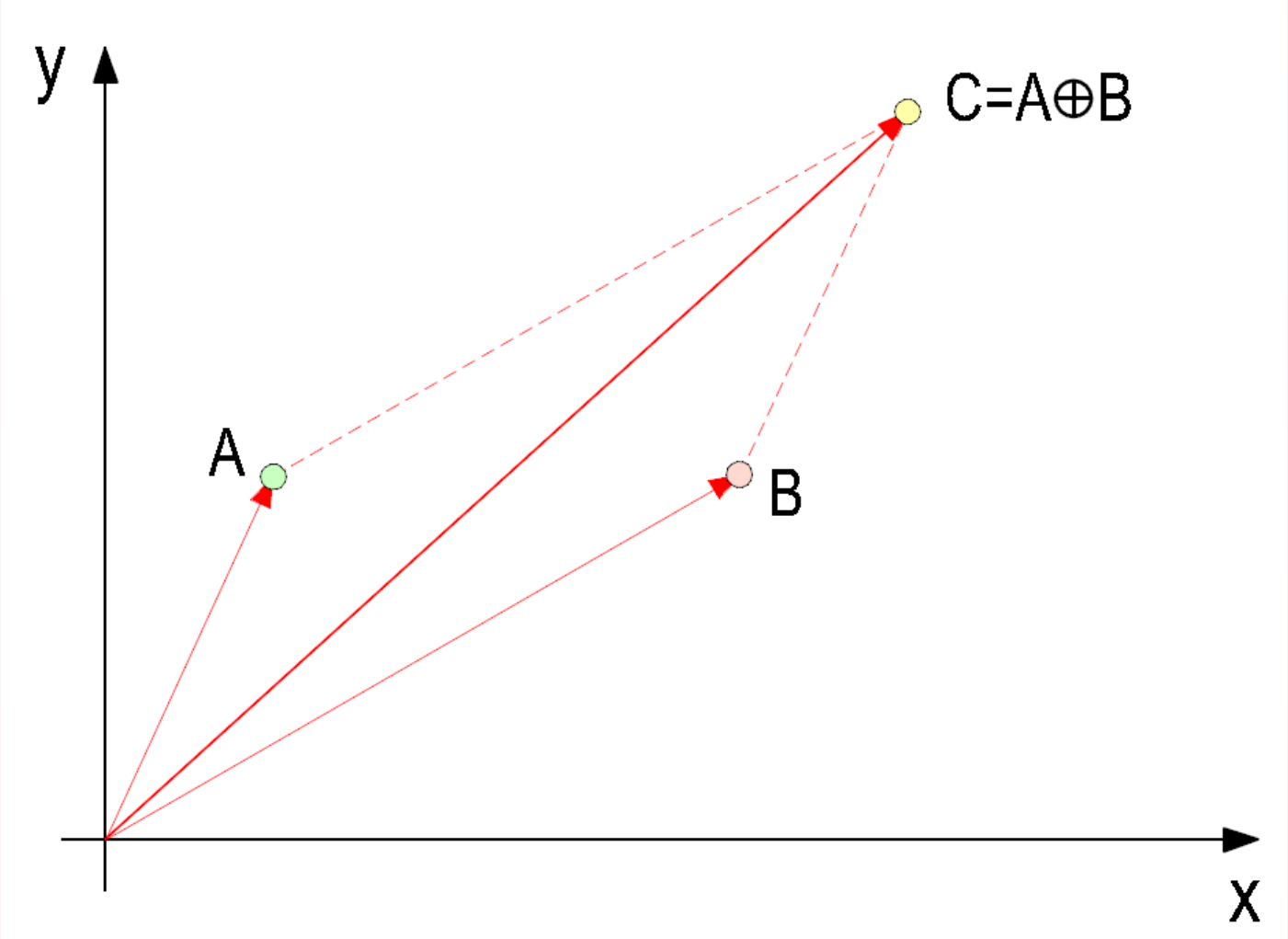
$$\mathcal{C}=\mathcal{A}^s \oplus \mathcal{B}^t = (\mathcal{A} \oplus \mathcal{B})^{s+t},$$

nemá vliv na velikost a tvar $\mathcal{C}$; $\mathcal{C}$ bude posunuta o součet vektorů $s + t$.



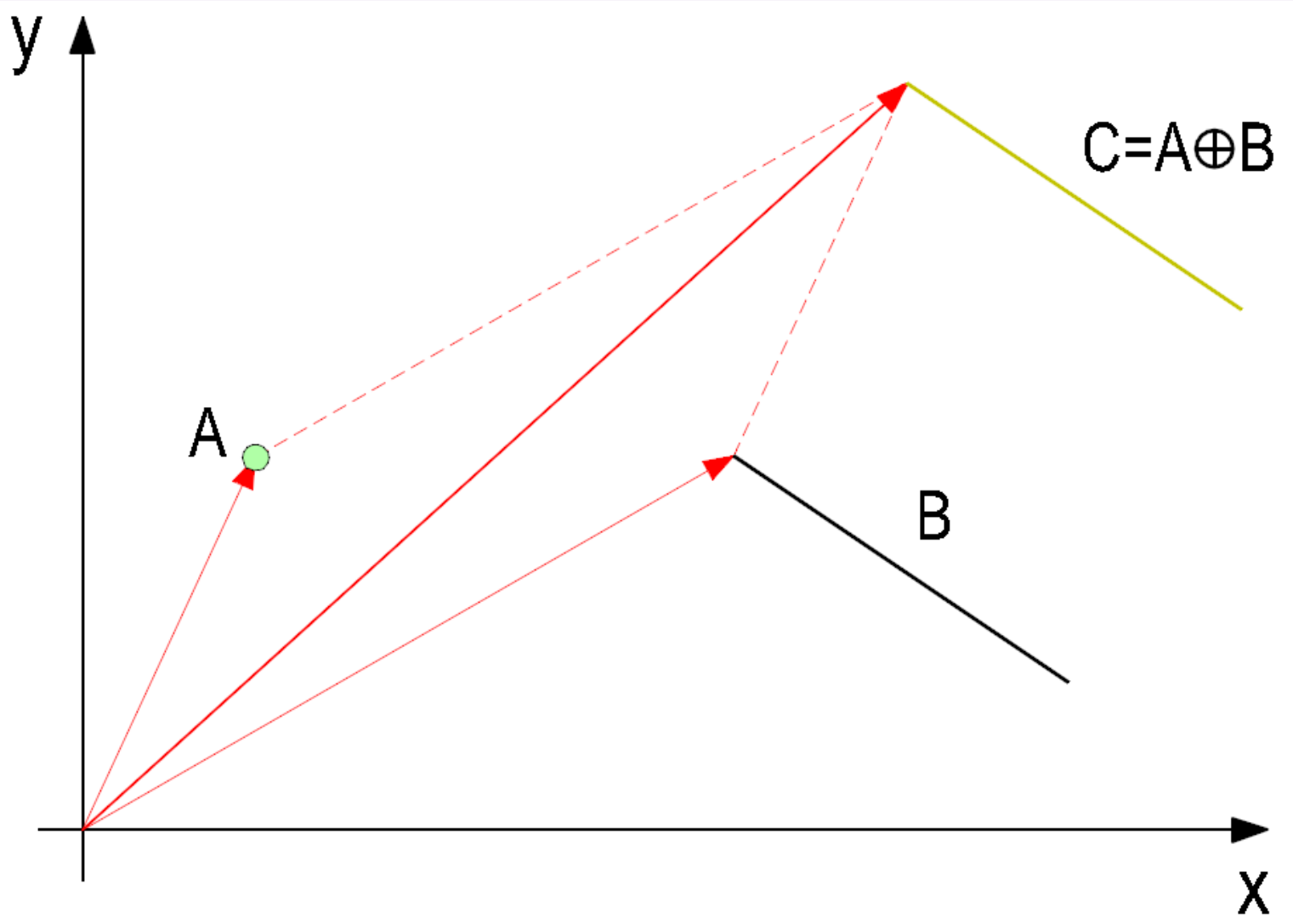
44. Minkowského suma 2 bodů

Minkowského suma dvou bodů je jednobodová množina.



45. Minkowského suma bodu a úsečky

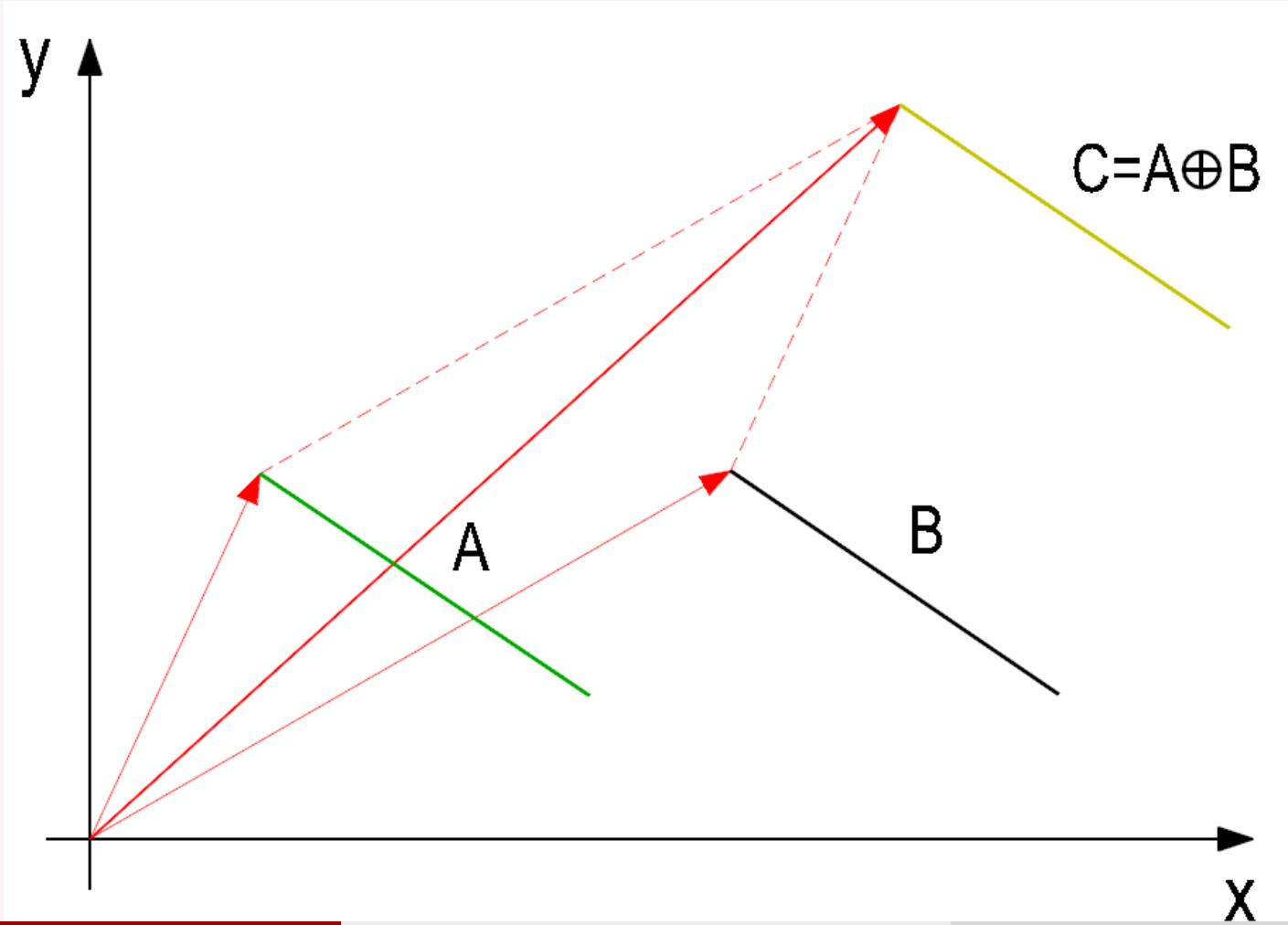
Minkowského suma bodu a úsečky jednobodová úsečka.



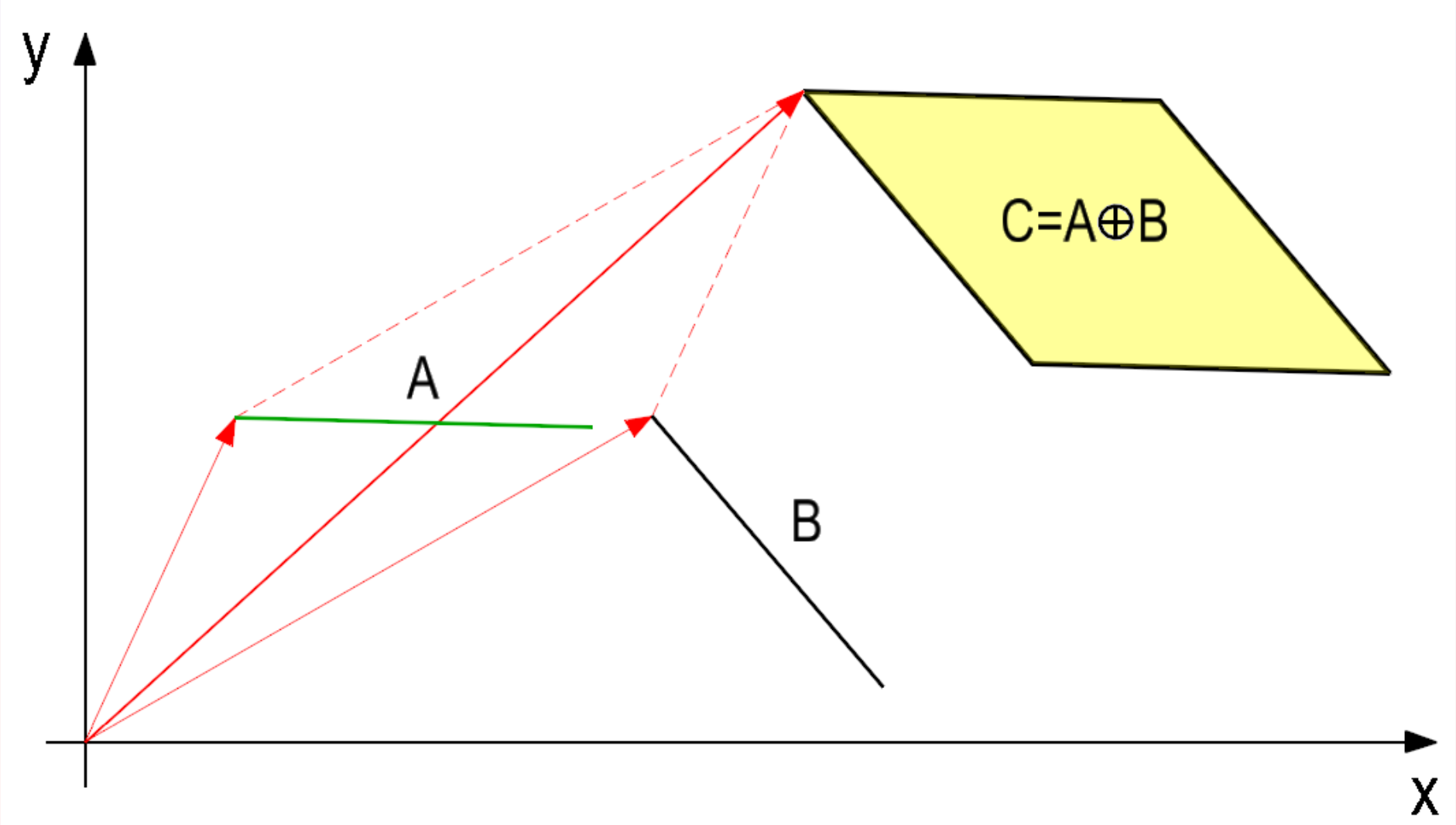
46. Minkowského suma 2 úseček

Pokud jsou úsečky rovnoběžné, je výsledkem úsečka.

V opačném případě výsledkem konvexní 4-úhelník.

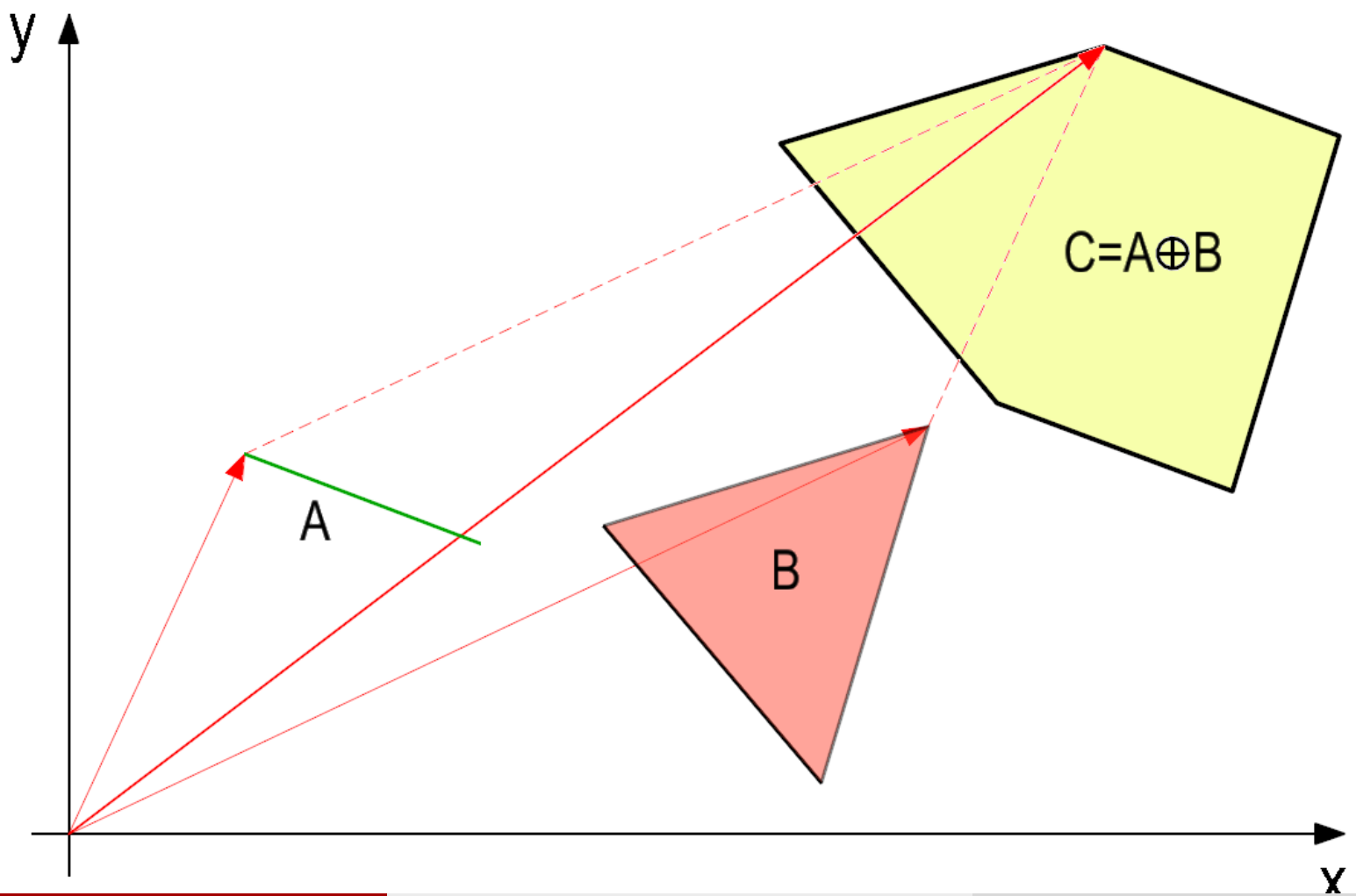


47. Minkowského suma 2 úseček



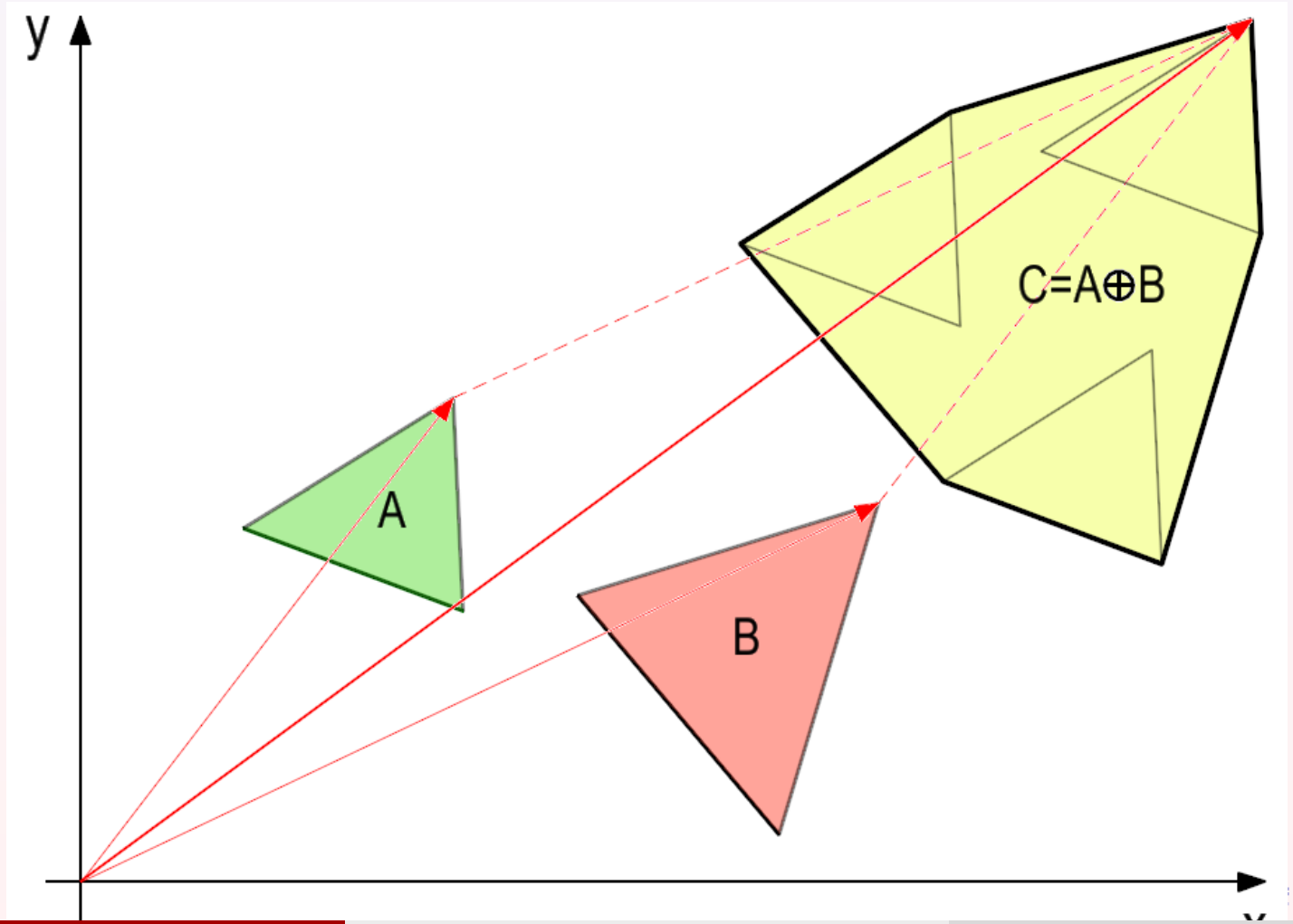
48. Minkowského úsečky a trojúhelníku

Pokud není úsečka rovnoběžná s žádnou stranou trojúhelníku, výsledkem je konvexní pětiúhelník.



49. Minkowského suma 2 trojúhelníků

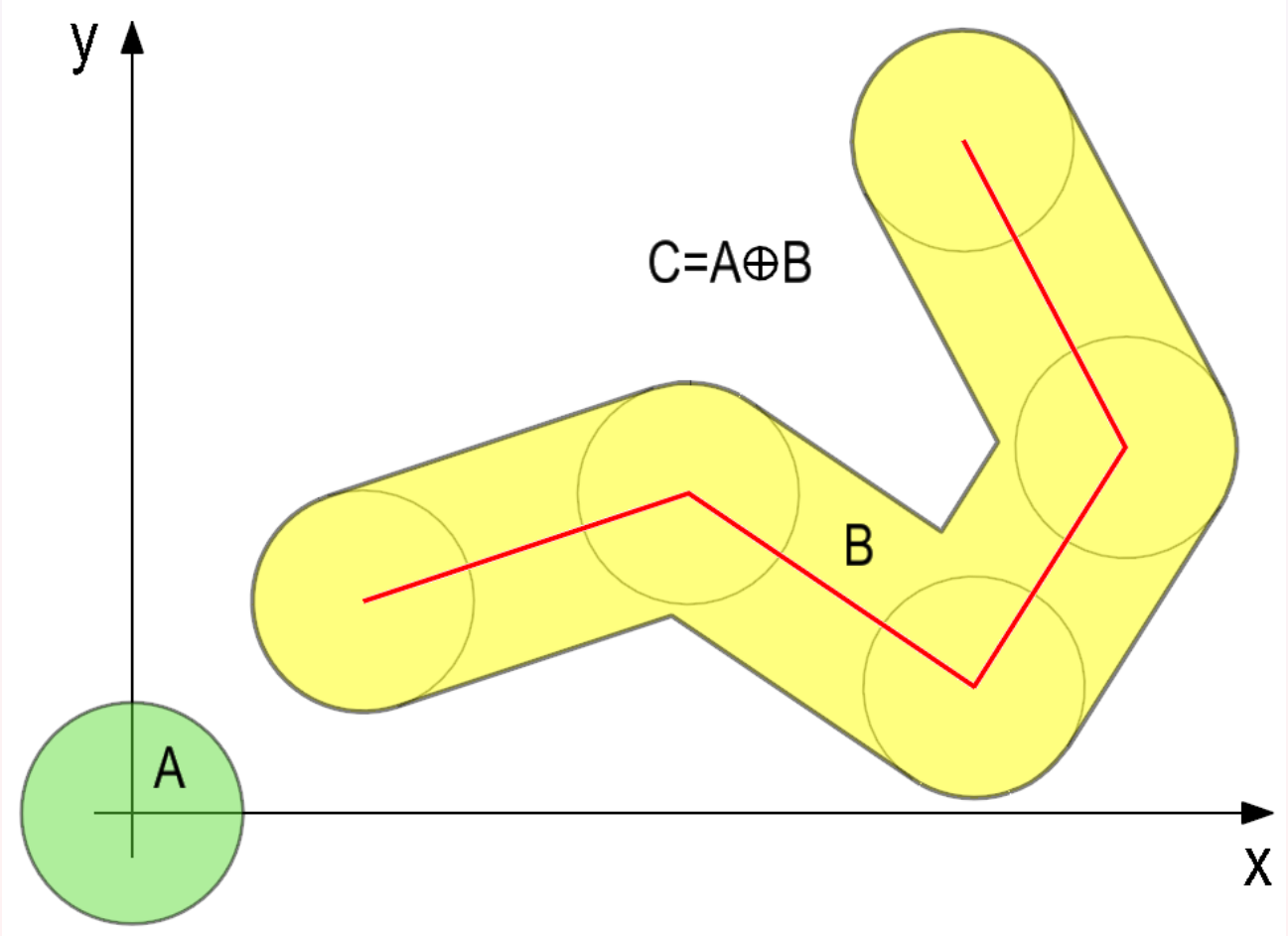
Minkowského suma dvou trojúhelníků představuje konvexní n úhelník s počty vrcholů $n \in (3, 6)$.



50. Minkowského suma linie a kruhu

Kruh k s poloměrem r, počátek v bodě $[0, 0]$.

Minkowského suma kruhu a linie L je offset L (buffer).



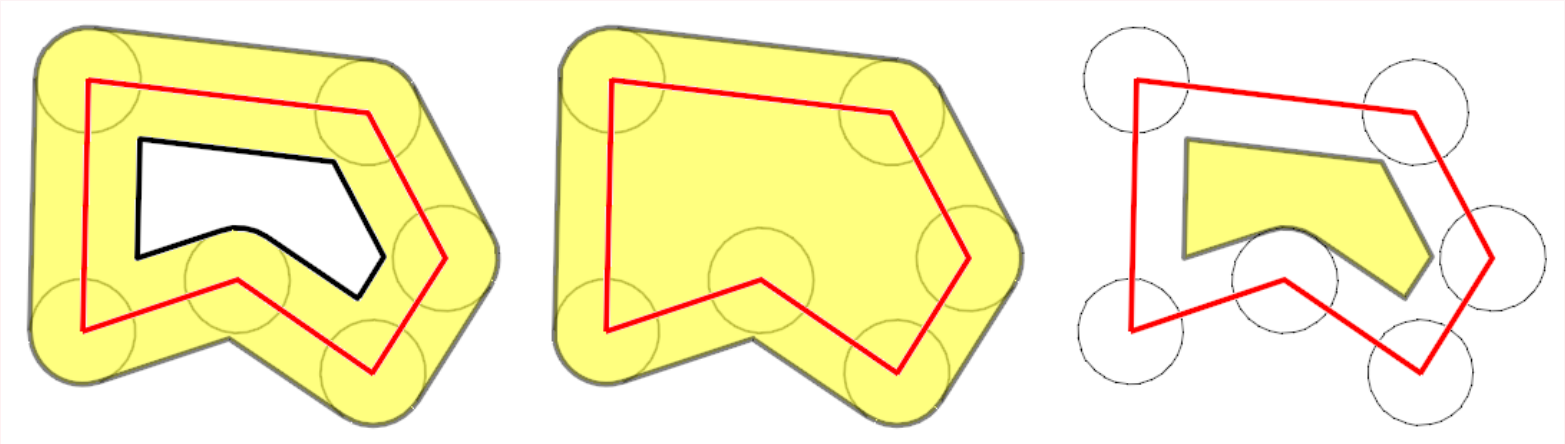
51. Minkowského suma uzavřené oblasti a kruhu

Kruh k s poloměrem r, počátek v bodě $[0, 0]$.

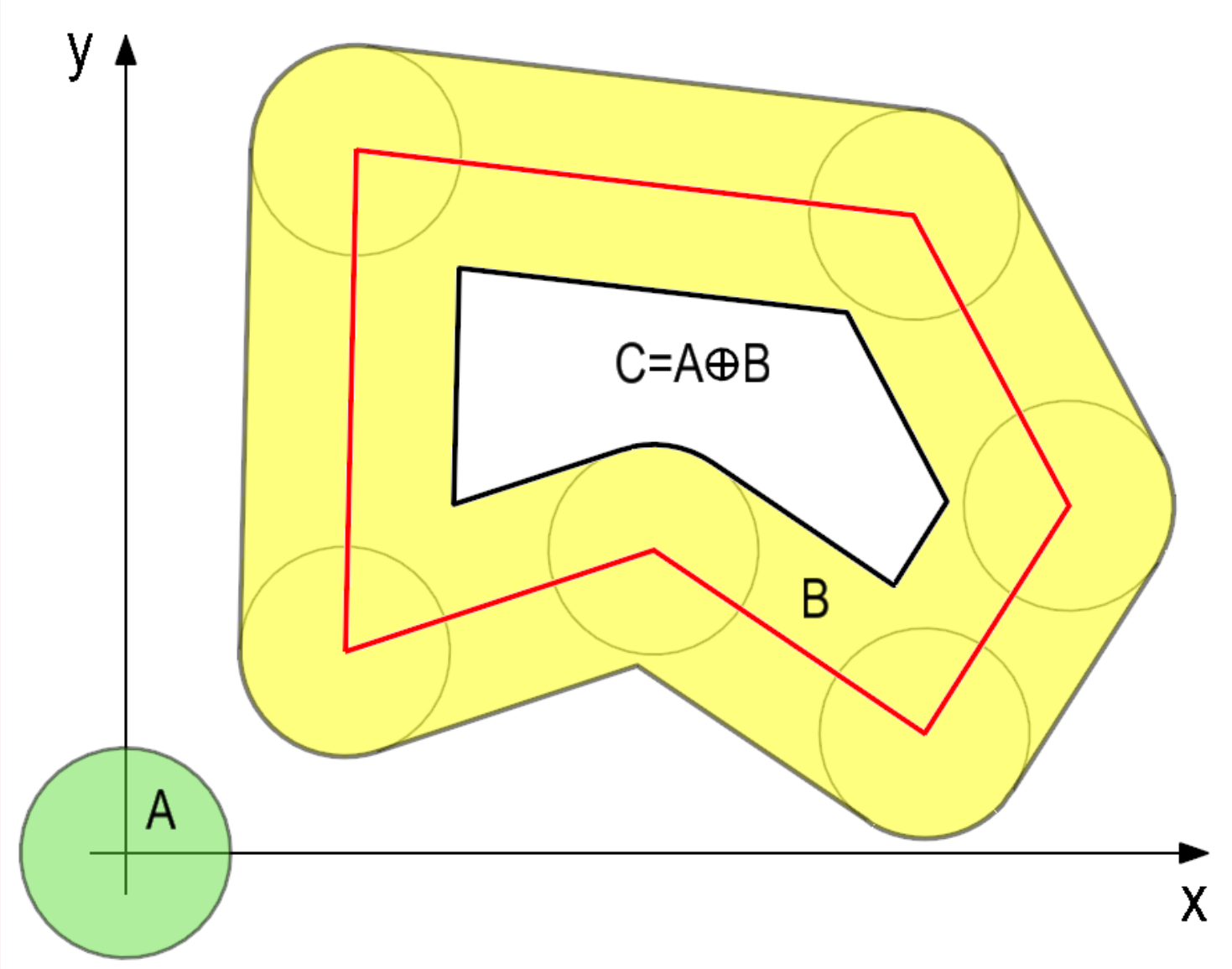
Minkowského suma kruhu a uzavřené oblasti C je offset C (buffer).

3 varianty bufferu nad uzavřenou oblastí C:

- *Vnitřní i vnější buffer:*
  Ekvidistanta konstruována uvnitř i vně C .
- *Vnější buffer*
  Ekvidistanta pouze vně C, nejčastěji používaná varianta.
- *Vnitřní buffer*
  Ekvidistanta pouze uvnitř C, nelze vždy zkonstruovat, problém singulárních segmentů.



52. Minkowského suma uzavřené oblasti a kruhu



53. Přehled algoritmů

Konstrukce Minkowského sumy:

- dekompozice na konvexní polygony,
- konvoluční metoda (složitý algoritmus).

Dekompoziční metoda.

Dekompozice oblasti na konvexní oblasti (triangulace).

Každá zpracována samostatně.

Extrémní bod množiny $\mathcal{C} = \mathcal{A} \oplus \mathcal{B}$ je součtem extrémních bodů množin $\mathcal{A}, \mathcal{B}$.

Hranici $\mathcal{C}$ tvoří pouze úsečky rovnoběžné s hranami $\mathcal{A}, \mathcal{B}$.

Následně sjednocení lokálních $\mathcal{C}$ nad oblastmi.

Princip algoritmu:

Nový bod c součtem dvou extrémních bodů

$$c = a_i + b_j,$$

Extrémní bod c je koncový bod "levějšího" z vektorů

$$u = a_{i+1} - a_i, \qquad v = b_{j+1} - b_j.$$

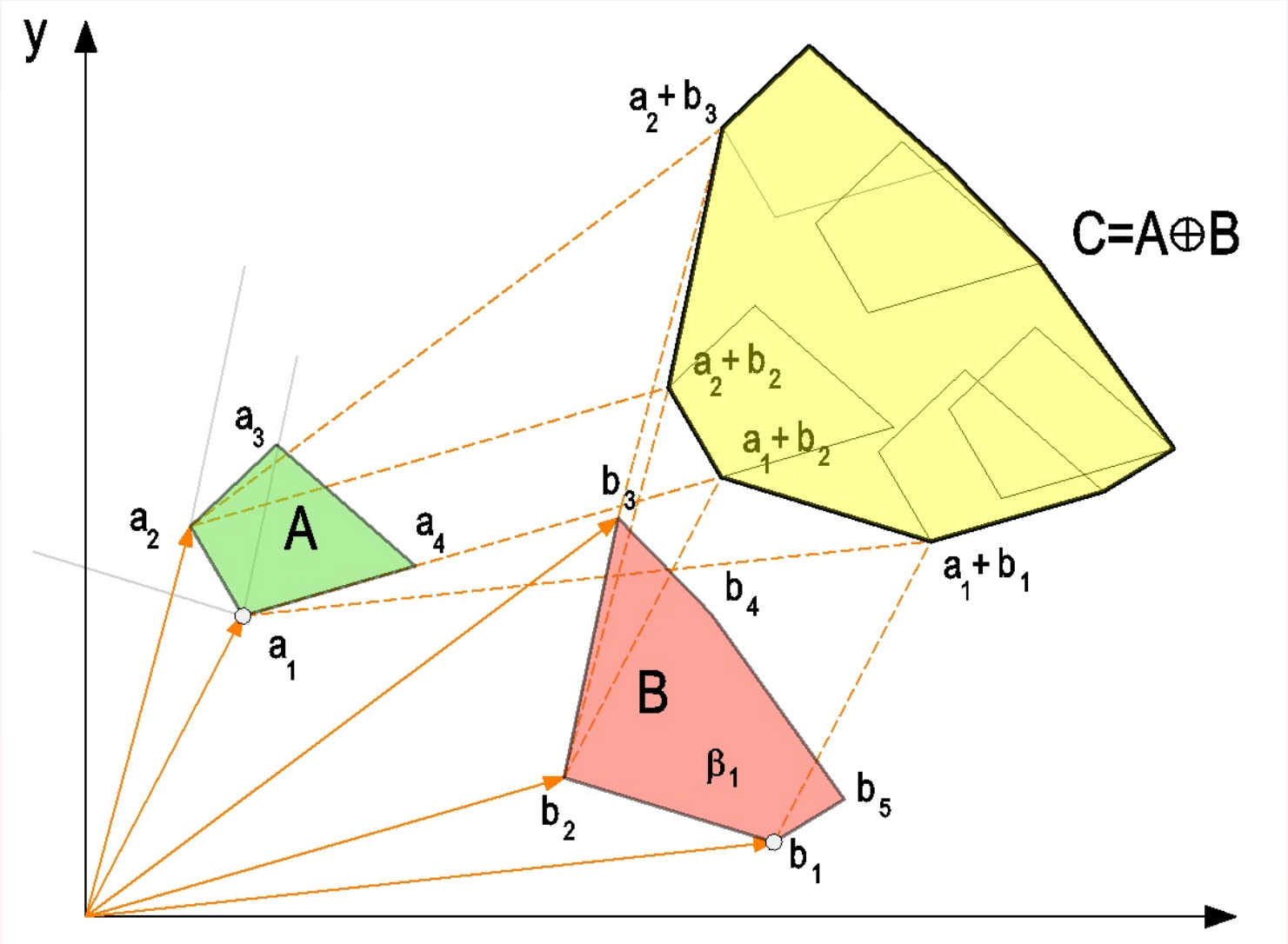
Spočteme orientaci v vzhledem k u

$$t = \begin{vmatrix} u_x & u_y \\ v_x & v_y \end{vmatrix}.$$

Inkrementace i, j dle výsledků testu (inkrementujeme index v "levějším" segmentu)

$$i = \begin{cases} i+1, & t \leq 0, \\ i, & \text{jinak}, \end{cases}, \qquad j = \begin{cases} j+1, & t \geq 0, \\ j, & \text{jinak}, \end{cases}.$$

54. Ilustrace algoritmu



55. Implementace algoritmu

CW orientace vrcholů oblasti.

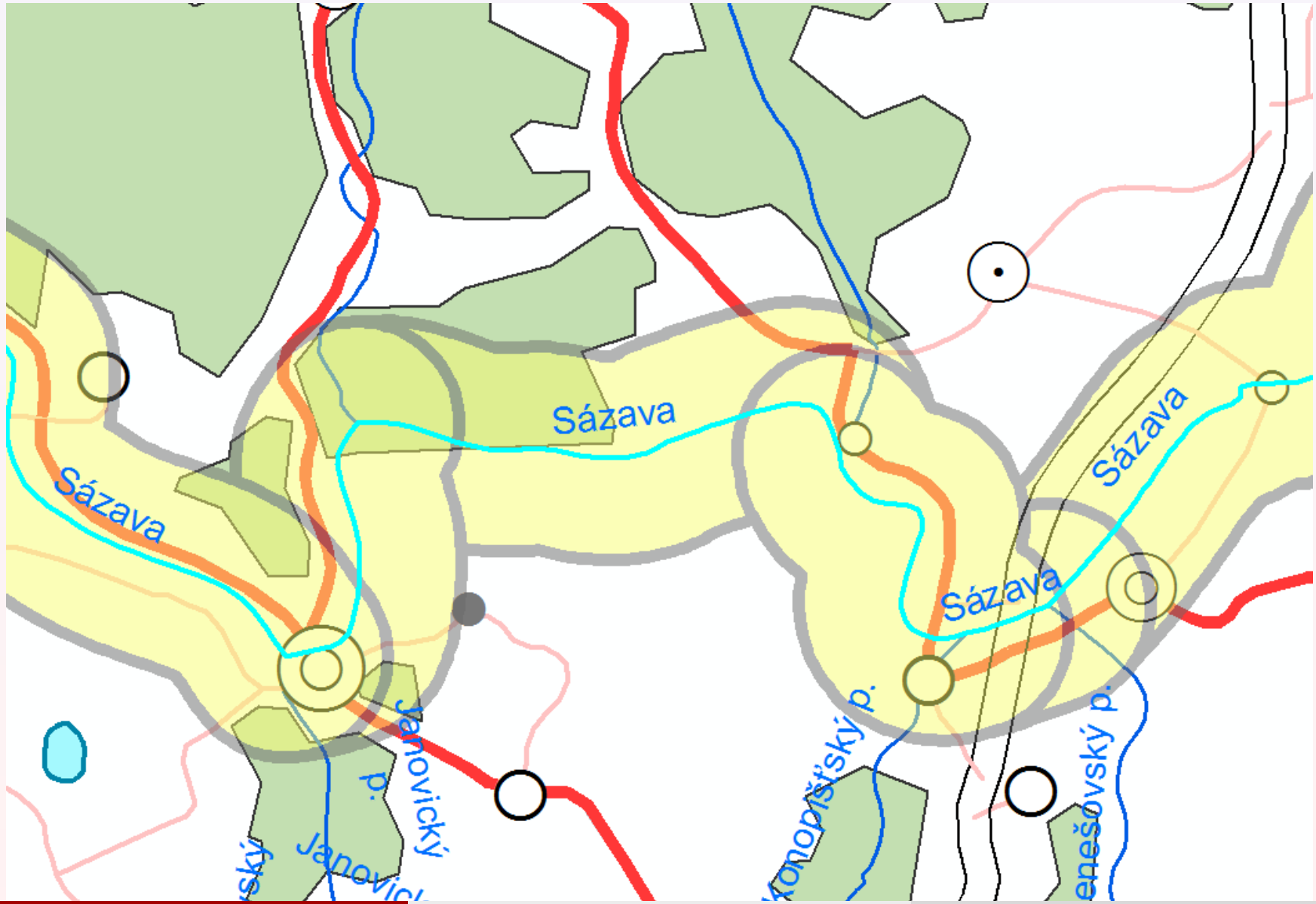
Složitost algoritmu $O(m+n)$.

Algoritmus 2: Minkowski $(\mathcal{A}, \mathcal{B}, C)$

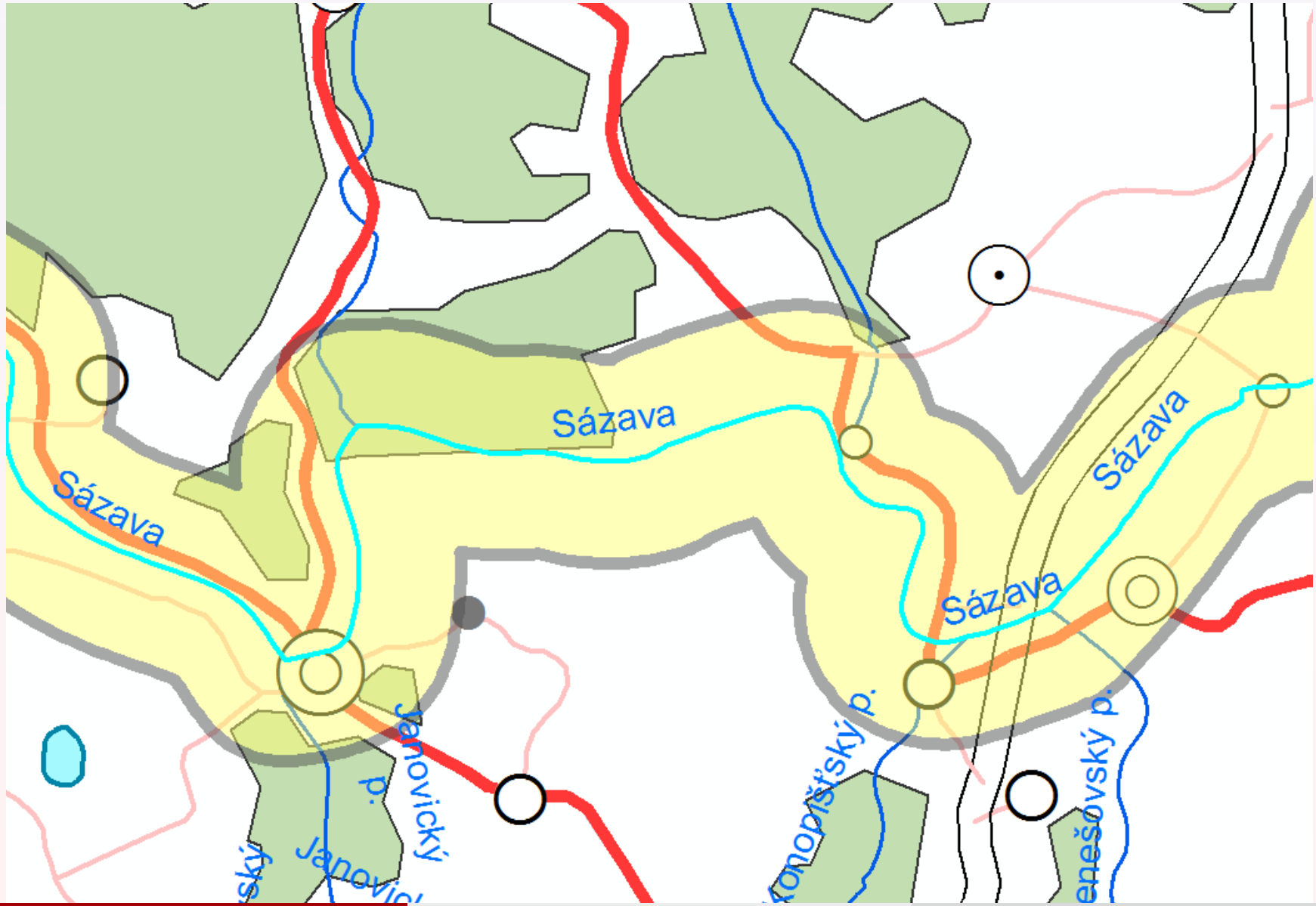
1: Seřazení vrcholů p_i, q_j oblastí $\mathcal{A}, \mathcal{B}$ ve směru hodinových ručiček.
2: Označ a_1 a b_1 extrémní vrcholy $\mathcal{A}, \mathcal{B}$ s min. hodnotou y.
3: Inicializuj $i, j, k = 1$.
4: while `i < len(`$\mathcal{A}$`)-2 or j < len(`$\mathcal{B}$`)-2`:
5: $c_k = a_i + b_j$, $k = k + 1$.
6: $u = a_{i+1} - a_i$, $v = b_{j+1} - b_j$
6: $t = \begin{vmatrix} u_x & u_y \\ v_x & v_y \end{vmatrix}$.
8: if $t <= 0$
9: $i = i + 1$
10: if $t >= 0$
11: $j = j + 1$

---

56. Ilustrace bufferu nad liniovým prvkem



57. Ilustrace bufferu nad liniovým prvkem (Dissolve)



58. Ilustrace bufferu nad uzavřenou oblastí